

NEUROCONTROLLER DEVELOPMENT FOR NONLINEAR PROCESSES UTILISING EVOLUTIONARY REINFORCEMENT LEARNING

Alex van Eck Conradie

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science (Chemical Engineering) at the University of Stellenbosch.



Supervisors: Professor C. Aldrich
Professor I. Nieuwoudt

Stellenbosch
South Africa
January 2000

DECLARATION

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and has not previously in its entirety or in part been submitted at any university for a degree.

Alex van Eck Conradie

24 January 2000

SYNOPSIS

NEUROCONTROLLER DEVELOPMENT FOR NONLINEAR PROCESSES UTILISING EVOLUTIONARY REINFORCEMENT LEARNING

The growth in intelligent control has primarily been a reaction to the realisation that nonlinear control theory has been unable to provide practical solutions to present day control challenges. Consequently the chemical industry may be cited for numerous instances of overdesign, which result as an attempt to avoiding operation near or within complex (often more economically viable) operating regimes. Within these complex operating regimes robust control system performance may prove difficult to achieve using conventional (algorithmic) control methodologies.

Biological neuronal control mechanisms demonstrate a remarkable ability to make accurate generalisations from sparse environmental information. Neural networks, with their ability to learn and their inherent massive parallel processing ability, introduce numerous opportunities for developing superior control structures for complex nonlinear systems. To facilitate neural network learning, reinforcement learning techniques provide a framework which allows for learning from direct interactions with a dynamic environment. Its promise as a means of automating the knowledge acquisition process is beguiling, as it provides a means of developing control strategies from cause and effect (reward and punishment) interaction information, without needing to specify how the goal is to be achieved.

This study aims to establish evolutionary reinforcement learning as a powerful tool for developing robust neurocontrollers for application in highly nonlinear process systems. A novel evolutionary algorithm; Symbiotic, Adaptive Neuro-Evolution (SANE), is utilised to facilitate neurocontroller development. This study also aims to introduce SANE as a means of integrating the process design and process control development functions, to obtain a single comprehensive calculation step for maximum economic benefit. This approach thus provides a tool with which to limit the occurrence of overdesign in the process industry.

To investigate the feasibility of evolutionary reinforcement learning in achieving these aims, the SANE algorithm is implemented in an event-driven software environment (developed in Delphi 4.0), which may be applied for both simulation and real world control problems. Four highly nonlinear reactor arrangements are considered in simulation studies. As a real world application, a novel batch distillation pilot plant, a Multi-Effect Batch Distillation (MEBAD) column, was constructed and commissioned.

The neurocontrollers developed using SANE in the complex simulation studies, were found to exhibit excellent robustness and generalisation capabilities. In comparison with model predictive control implementations, the neurocontrollers proved far less sensitive to model parameter uncertainties, removing the need for model mismatch compensation to eliminate steady state off-set. The SANE algorithm also proved highly effective in discovering the operating region of greatest economic return, while simultaneously developing a neurocontroller for this optimal operating point. SANE, however, demonstrated limited success in learning an effective control policy for the MEBAD pilot plant (poor generalisation), possibly due to limiting the algorithm's search to a too small region of the state space and the disruptive effects of sensor noise on the evaluation process.

For industrial applications, starting the evolutionary process from a random initial genetic algorithm population may prove too costly in terms of time and financial considerations. Pretraining the genetic algorithm population on approximate simulation models of the real process, may result in an acceptable search duration for the optimal control policy. The application of this neurocontrol development approach from a plantwide perspective should also have significant benefits, as individual controller interactions are so doing implicitly eliminated.

OPSOMMING

NEUROBEHEER ONTWIKKELING VIR NIE-LINIÊRE PROSESSE MET EVOLUTIONÊRE VERSTERKINGSLEER

The huidige groei in intelligente beheerstelsels is primêr 'n reaksie op die besef dat nie-liniêre beheerstelsel teorie nie instaat is daartoe om praktiese oplossings te bied vir huidige beheer kwelkwessies nie. Gevolglik kan talle insidente van oorontwerp in die chemiese nywerhede aangevoer word, wat voortvloei uit 'n poging om bedryf in of naby komplekse bedryfsgebiede (dikwels meer ekonomies vatbaar) te vermy. Die ontwikkeling van robuuste beheerstelsels, met konvensionele (algoritmiese) beheertegnieke, in die komplekse bedryfsgebiede mag problematies wees.

Biologiese neurobeheer meganismes vertoon 'n merkwaardige vermoë om te veralgemeen vanaf yl omgewingsdata. Neurale netwerke, met hulle vermoë om te leer en hulle inherente parallelle verwerkingsvermoë, bied talle geleenthede vir die ontwikkeling van meer doeltreffende beheerstelsels vir gebruik in komplekse nie-liniêre sisteme. Versterkingsleer bied a raamwerk waarbinne 'n neurale netwerk leer deur direkte interaksie met 'n dinamiese omgewing. Versterkingsleer hou belofte in vir die inwin van kennis, deur die ontwikkeling van beheerstrategieë vanaf aksie en reaksie (loon en straf) interaksies - sonder om te spesifiseer hoe die taak voltooi moet word.

Hierdie studie beaam om evolutionêre versterkingsleer as 'n kragtige strategie vir die ontwikkeling van robuuste neurobeheerders in nie-liniêre prosesomgewings, te vestig. 'n Nuwe evolutionêre algoritme; Simbiotiese, Aanpasbare, Neuro-Evolusie (SANE), word aangewend vir die ontwikkeling van die neurobeheerders. Hierdie studie beoog ook die daarstelling van SANE as 'n weg om prosesontwerp en prosesbeheer ontwikkeling vir maksimale ekonomiese uitkering, te integreer. Hierdie benadering bied dus 'n strategie waardeur die insidente van oorontwerp beperk kan word.

Om die haalbaarheid van hierdie doelwitte, deur die gebruik van evolusionêre versterkingsleer te ondersoek, is die SANE algoritme aangewend in 'n Windows

omgewing (ontwikkel in Delphi 4.0). Die Delphi programmatuur geniet toepassing in beide die simulاسie en werklike beheer probleme. Vier nie-liniêre reaktore ontwerpe is oorweeg in die simulاسie studies. As 'n werklike beheer toepassing, is 'n nuwe enkelladingsdistillasie kolom, 'n Multi-Effek Enkelladingskolom (MEBAD) gebou en in bedryf gestel.

Die neurobeheerders vir die komplekse simulاسie studies, wat deur SANE ontwikkel is, het uitstekende robuustheid en veralgemeningsvermoë ten toon gestel. In vergelyking met model voorspellingsbeheer implementasies, is gevind dat die neurobeheerders heelwat minder sensitief is vir model parameter onsekerheid. Die noodsaak na model onsekerheid kompensاسie om gestadigde toestand afset te elimineer, word gevolglik verwyder. The SANE algoritme is ook hoogs effektief vir die soek na die mees ekonomies bedryfstoestand, terwyl 'n effektiewe neurobeheerder gelyktydig vir hierdie ekonomies optimumgebied ontwikkel word. SANE het egter beperkte sukses in die leer van 'n effektiewe beheerstrategie vanaf die MEBAD toetsaanleg getoon (swak veralgemening). Die swak veralgemening kan toegeskryf word aan 'n te klein bedryfsgebied waarin die algoritme moes soek en die negatiewe effek van sensor geraas op die evaluاسie proses.

Vir industriële applikasies blyk dit dat die uitvoer van die evolutionêre proses vanaf 'n wisselkeurige begintoestand nie koste effektief is in terme van tyd en finansies nie. Deur die genetiese algoritme populasie vooraf op 'n benaderde model op te lei, kan die soek tydperk na 'n optimale beheerstrategie aansienlik verkort word. Die aanwending van die neurobeheer ontwikkelingstrategie vanuit 'n aanlegwye oogpunt mag aanleiding gee tot aansienlike voordele, aangesien individuele beheerder interaksies sodoende implisiet uitgeskakel word.

ACKNOWLEDGEMENTS

My Father in heaven, for being gracious to me.

My supervisors, Professor Izak Nieuwoudt and Professor Chris Aldrich; for their support, for making me a better engineer, for being patient with youthful enthusiasm and for believing in me.

The workshop staff, particularly Anton and Vincent, for their skill and aid.

Friends and family who encouraged, loved and laughed with me.

Academics in the reinforcement learning and evolutionary algorithm business, who through their research showed me the beauty of their art.

Loving thanks to my wife, Anje, who braved the challenges of student life with me once again.

*In remembrance of my beloved godmother, Ronnie Rees,
who watches over me still.*

CONTENTS

Synopsis	i
Opsomming	iii
Acknowledgements & Dedication	v

CONTENTS	vii
----------	-----

LIST OF TABLES	xiii
----------------	------

LIST OF FIGURES	xiii
-----------------	------

INTRODUCTION	1
--------------	---

1.1 MOTIVATION FOR NONLINEAR CHEMICAL PROCESS CONTROL	1
--	---

1.2 OVERDESIGN TO AVOID COMPLEX NONLINEARITIES	2
--	---

1.3 CONVENTIONAL AND NEURAL DESIGN METHODOLOGIES	3
1.3.1 Algorithmic and biological approaches to information processing	3
1.3.2 Conventional Control System Design	5
1.3.3 Neural Controller Design	7

1.4 REINFORCEMENT LEARNING FOR NEURAL CONTROLLER DESIGN	8
--	---

1.5 OBJECTIVES OF THIS STUDY	9
------------------------------	---

2 REINFORCEMENT LEARNING	11
--------------------------	----

2.1 ELEMENTS OF REINFORCEMENT LEARNING	11
--	----

2.2 REINFORCEMENT LEARNING VS. SUPERVISED LEARNING	12
--	----

2.3 THE AGENT-ENVIRONMENT INTERFACE	13
-------------------------------------	----

2.4 MODELS OF OPTIMAL BEHAVIOUR	15
---------------------------------	----

2.5 CREDIT ASSIGNMENT	17
-----------------------	----

2.6 THE MARKOV ASSUMPTION	18
2.6.1 The Markov Property	18
2.6.2 The ubiquity of non-Markov tasks	19
2.6.3 Difficulties for reinforcement learning	20

2.7 EXPLORATION AND EXPLOITATION	21
----------------------------------	----

2.8	GENERALISATION	23
2.9	CONTROL POLICY REVISIONS AND MEMORY	24
2.10	CONCLUDING REMARKS	25
2.11	SYMBOLS FOR CHAPTER 2	26
3	EVOLVING NEURAL NETWORKS FOR CONTROL	27
3.1	FOUNDATIONS OF GENETIC ALGORITHMS	27
3.2	DIFFICULTIES IN EVOLVING NEURAL NETWORKS	30
3.3	MAINTAINING DIVERSITY	32
3.3.1	Restricting the selection process (preselection & crowding models)	34
3.3.2	Dividing the population into subpopulations	35
3.3.3	Restricting the mating procedure (local mating)	36
3.3.4	Explicit fitness sharing	37
3.3.5	Implicit fitness sharing	39
3.4	COMPARING THE SANE APPROACH TO OTHER NEURAL NETWORK APPROACHES	40
3.5	MAINTAINING EFFECTIVE NEURON COLLECTIONS	44
3.6	SANE IMPLEMENTATION	45
3.6.1	Evaluation stage	48
3.6.2	Recombination for the neuron population	50
3.6.3	Recombination for the network population	51
3.7	THE SUITABILITY OF SANE FOR EVOLVING NEURAL NETWORKS	52
3.8	PAST APPLICATIONS OF SANE	53
3.8.1	Game Tree Search	53
3.8.2	Controlling Chaos	53
3.8.3	Robot Arm Control	55
3.9	REINFORCEMENT LEARNING AS A STRATEGY TO PROCESS CONTROL DESIGN	56
3.10	CONCLUDING REMARKS	58
3.11	SYMBOLS FOR CHAPTER 3	60
4	NEURAL NETWORKS FOR CONTROL SYSTEMS	61
4.1	NEURAL NETWORK CONTROL STRUCTURES	61
4.1.1	Supervised Control	63
4.1.2	Direct Inverse Control	63

4.1.3	Model Reference Control	64
4.1.4	Internal Model Control (IMC)	65
4.1.5	Model Predictive Control	66
4.2	CONCLUDING REMARKS FOR CHAPTER 4	69
4.3	SYMBOLS FOR CHAPTER 4	70
PROCESS CONTROL NEUROCONTROL SIMULATION CASE STUDIES		71
5.1	NEURAL PROCESS CONTROL UTILIZING SANE	71
5.1.1	SANE fitness function implementation	71
5.1.2	SANE Control Structure Implementation	73
5.2	A BENCHMARK BIOREACTOR CONTROL PROBLEM	74
5.2.1	Bioreactor Process Description	74
5.2.2	Bioreactor Control with a Multi-step Nonlinear Predictive Controller	78
5.2.3	Bioreactor control with Kalman Filter Trained Recurrent Networks	81
5.2.4	Neurocontrol using a simplification of the Backpropagation-Through-Time-Algorithm	83
5.2.5	Bioreactor control utilising functional expansion models	84
5.2.6	NeuroControl Workbench Results	86
5.2.7	Neural reinforcement control using ELEGANCE	87
5.2.8	Coevolutionary Bioreactor Reinforcement Learning implementation	88
5.2.9	Concluding remarks for control strategies considered in previous research	89
5.2.10	Symbiotic, Adaptive Neuro Evolution (SANE)	90
5.2.10.1	<i>Learning optimal behaviour</i>	90
5.2.10.2	<i>Rejecting Nonminimum Phase Behaviour</i>	92
5.2.10.3	<i>Improving the dynamic response of the system</i>	95
5.2.10.4	<i>Servo Response Characteristics</i>	98
5.2.10.5	<i>Plant / Model Mismatch</i>	102
5.2.10.6	<i>Unmeasured Disturbance Rejection</i>	107
5.2.10.7	<i>Controller performance with Sensor Noise</i>	109
5.2.11	Process Design and Control Optimisation Integration	112
5.3	CATALYTIC REACTOR SIMULATION	114
5.3.1	Catalytic Reactor Process Description	114
5.3.2	Single Step pseudo-Newton Model Predictive Control algorithm	118
5.3.3	Multi-Step Nonlinear Predictive Controller	120
5.3.4	Nonlinear Quadratic Dynamic Matrix Control with State Estimation	121
5.3.5	Concluding remarks for control strategies considered in previous research	123
5.3.6	Symbiotic, Adaptive Neuro Evolution (SANE)	123
5.3.6.1	<i>Learning optimal behaviour</i>	125

5.3.6.2	<i>Servo Characteristics</i>	127
5.3.6.3	<i>Large Feed Concentration Disturbance</i>	133
5.4	NONISOTHERMAL REACTOR SIMULATION	137
5.4.1	Nonisothermal Reactor Process Description	137
5.4.2	Single Step pseudo-Newton Model Predictive Control algorithm	141
5.4.3	Nonlinear Model Predictive Control Using Second-Order Model Approximation	142
5.4.3.1	<i>Servo characteristics</i>	142
5.4.3.2	<i>Model / Plant mismatch analysis</i>	146
5.4.4	Direct and Indirect Model Based Control using ANN	147
5.4.5	Concluding remarks for control strategies considered in previous research	149
5.4.6	Symbiotic, Adaptive Neuro Evolution (SANE)	150
5.4.6.1	<i>Neurocontroller evaluation learning</i>	150
5.4.6.2	<i>Servo Response Characteristics</i>	152
5.4.6.3	<i>Robustness analysis of the SANE direct method approach</i>	157
5.4.6.4	<i>Model Parameter Perturbation analysis</i>	158
5.5	VAN DE VUSSE REACTOR	162
5.5.1	Process Description	162
5.5.2	Control Policies cited for the nonminimum phase reactor	166
5.5.3	Concluding remarks for control strategies considered in previous research	167
5.5.4	Symbiotic, Adaptive Neuro Evolution (SANE)	168
5.5.4.1	<i>Neurocontroller evaluation learning</i>	168
5.5.4.2	<i>Servo response characteristics</i>	171
5.5.4.3	<i>Feed concentration disturbance rejection</i>	174
5.5.4.4	<i>Disturbance rejection in multiple simultaneously varying model parameters</i>	177
5.5.4.5	<i>Increased economic benefit through unconstrained controller development</i>	180
5.6	CONCLUDING REMARKS FOR CHAPTER 5	182
5.7	SYMBOLS FOR CHAPTER 5	183
MULTI-EFFECT BATCH DISTILLATION – REAL WORLD APPLICATION		186
6.1	PROCESS DESCRIPTION	187
6.1.1	Batch Distillation in Continuous Distillation Columns	187
6.1.2	Multi-Effect Batch Distillation Column	190
6.2	PROCESS APPLICATIONS	192
6.2.1	Separation of ideal multi-component mixtures	192
6.2.2	Extractive Batch Distillation	194
6.2.3	Reactive Batch Distillation	195
6.3	ECONOMIC POTENTIAL OF THE MULTI-VESSEL COLUMN	197

6.3.1	Conventional batch distillation operating strategies	197
6.3.2	Multi-effect batch Distillation	198
6.4	OPERATION AND CONTROL POLICIES FOR THE MEBAD CONFIGURATION	199
6.4.1	MEBAD total reflux operation with vessel hold-up control	200
6.4.2	Temperature control feedback structure	202
6.4.2.1	<i>Experimental verification</i>	202
6.5	EXPERIMENTAL	204
6.5.1	Experimental Set-up	204
6.5.1.1	<i>Mechanical Equipment</i>	204
6.5.1.2	<i>Instrumentation and signal processing</i>	206
6.5.1.3	<i>Electrical Equipment</i>	207
6.5.1.4	<i>Man Machine Interface (MMI)</i>	208
6.5.2	Experimental Procedure	209
6.5.2.1	<i>PI control experimental procedure</i>	209
6.5.2.2	<i>Neurocontroller learning experimental procedure</i>	211
6.6	DISCUSSION OF RESULTS	216
6.6.1	Temperature PI control for MEBAD column	216
6.6.2	Neurocontroller performance controlling MEBAD column	218
6.7	CONCLUDING REMARKS FOR CHAPTER 6	223
6.8	SYMBOLS FOR CHAPTER 6	224
7	CONCLUSIONS AND RECOMMENDATION FOR FUTURE WORK	225
7.1	CONCLUSIONS	225
7.2	RECOMMENDATION FOR FUTURE WORK	226
	REFERENCES	229
	APPENDIX A	237
9.1	THE INVERTED PENDULUM PROBLEM	237
9.2	A PERFORMANCE BASED FORMULATION OF THE INVERTED PENDULUM PROBLEM	240
9.3	MODERN LINEAR CONTROL AND NEUROCONTROL SOLUTIONS TO THE INVERTED PENDULUM PROBLEM	241
9.3.1	Development of the neurocontroller	241
9.3.2	Development of the linear controller	244
9.4	COMPARATIVE AND SENSITIVITY ANALYSIS FOR CONTROLLER ROBUSTNESS	246

9.4.1	Linear controller performance in the nonlinear plant control loop	246
9.4.2	Comparative controller performance under plant/model mismatches	248
9.4.3	Comparative Control Performance in the presence of disturbances	252
9.5	SANE LEARNING ROBUSTNESS	254
9.5.1	Non-Markov Robustness Analysis	254
9.5.2	Learning Ability in a Noisy Environment	257
9.6	CONCLUDING REMARKS	259
9.7	SYMBOLS FOR APPENDIX A	260
10	APPENDIX B - SOURCE CODE DESCRIPTION	261
10.1	SOURCE CODE COMMON TO SIMULATION STUDIES AND REAL WORLD APPLICATION	261
10.1.1	Best_Nets.pdf	261
10.1.2	PointNeuronPop.pdf	261
10.1.3	Sane_Config.pdf	261
10.1.4	Sane_Domain.pdf	261
10.1.5	Sane_EA.pdf	261
10.1.6	Sane_Network.pdf	262
10.1.7	Sane_Networkpop.pdf	262
10.1.8	Sane_Neuron.pdf	262
10.1.9	Sane_Neuronpop.pdf	262
10.1.10	Sane_NN.pdf	262
10.1.11	Sane_Util.pdf	262
10.1.12	SaneAlgThread.pdf	263
10.2	SOURCE CODE UNIQUE TO MEBAD MAN_MACHINE INTERFACE AND SANE ON-LINE LEARNING IMPLEMENTATION	263
10.2.1	Access.pdf	263
10.2.2	AdvanAO.pdf	263
10.2.3	AIFilter.pdf	264
10.2.4	BestNetThread.pdf	264
10.2.5	Driver.pdf	264
10.2.6	Global.pdf	264
10.2.7	MantleControl.pdf	264
10.2.8	PIDControl.pdf	264
10.3	SOURCE CODE UNIQUE TO SIMULATION STUDIES	264
10.3.1	Cstr.pdf / Bioreact.pdf / Pole.pdf	265
11	APPENDIX C - GLOSSARY	266

LIST OF TABLES

Table 1-1 - Common Process Control Design Complications	3
Table 3-1 - Fitness sharing limitations	38
Table 4-1 - Neural networks in control (Stephanopoulos & Han, 1996)	61
Table 4-2 - Neural network characteristics relevant to control systems (Kim et al., 1997)	62
Table 5-1 - Model parameter formulation for the Agrawal bioreactor	76
Table 5-2 - Harris et al. (1998) nominal operating steady state for the bioreactor.	84
Table 5-3 - Model parameters for the dynamic model of the isothermal slurry reactor.	116
Table 5-4 - Model parameter description and formulation for the nonisothermal reactor	139
Table 5-5 - Initial conditions considered by Li & Biegler (1988)	141
Table 5-6 - Model parameter formulations for the Van de Vusse reactions	165
Table 6-1 - Reactive batch distillation comparative study	195
Table 6-2 - Conditions for initialising the cascade control scheme	201
Table 6-3 - MEBAD column temperature set points	209
Table 9-1 - Model parameter formulation for the inverted pendulum	239

LIST OF FIGURES

Figure 2-1 - Agent-Environment Interaction	14
Figure 3-1 - Hyperplane Cube	29
Figure 3-2 - Structural / Functional Mapping	31
Figure 3-3 - An illustration of the neuro-evolution performed in SANE (b) compared to the standard approach to neuro-evolution (a).	42
Figure 3-4 - Network blueprint population in relation to the neuron population.	45
Figure 3-5 - Macro flow chart of the SANE algorithm	47
Figure 3-6 - Micro flow chart for evaluation phase for a single individual	49
Figure 4-1 - Direct Inverse Control	64
Figure 4-2 - Direct model reference control	65
Figure 4-3 - Structure for internal model control	65
Figure 4-4 - Model Predictive Control structure utilising neural networks	69
Figure 5-1 - Final Control structure for SANE neurocontroller	73
Figure 5-2 - Schematic diagram of the bioreactor	75
Figure 5-3 - Bioreactor Multiple Steady States - Stable (—), Unstable (---) and Periodic (·) steady states of cell mass as a function of Damkohler number for the Agrawal bioreactor with $\gamma = 0.48$ [] and $\beta = 0.02$ [] (Bregel & Seider, 1989).	77
Figure 5-4 - Transient response to step change in C_1^{SP} with a prediction horizon of $P = 4$. X_1 represents a deviation variable from the nominal set point ($C_1^{SP} = 0.1707$ []) (Bregel & Seider, 1989).	79
Figure 5-5 - Transient response to step change in C_1^{SP} with model mismatch in γ (Bregel & Seider, 1989)	80
Figure 5-6 - Transient response to step change in C_1^{SP} with process / mismatch. The steady state correction through mismatch compensation is illustrated (Bregel & Seider, 1989).	81
Figure 5-7 - Neural network model reference control for the nominal plant parameters with full-state information. The set point is plotted as a dashed line (Puskorius & Feldkamp, 1994).	82
Figure 5-8 - Performance of the controller network for a bioreactor characterised by a 5% change in γ and 20% in β and added measurement noise. The dashed lines represent the reference signal; the measurement noise reaching the system through the controller (Puskorius & Feldkamp, 1994).	83
Figure 5-9 - Closed loop behaviour of the bioreactor for a set point change from SP(0.0965, 0.912) to SP(0.1265, 0.8720). x_1 is the cell mass concentration deviation from the initial set point. u is the flow rate change from the initial steady state (····, linear model), (----, FEx model), (—, numerical) (Harris & Palazoglu, 1998).	85
Figure 5-10 - Closed loop behaviour for a set point change from SP(0.0965, 0.912) to SP(0.0785, 0.9345). x_1 is the cell mass concentration deviation from the initial set point. u is the flow rate change from the initial steady state (····, linear model), (----, FEx model), (—, numerical) (Harris & Palazoglu, 1998).	86
Figure 5-11 - NeuroControl Workbench control system servo response (C_1, C_2), for set point changes between (0.1207, 0.88) and (0.2107, 0.7226) (Jarmulak et al., 1997)	87

Figure 5-12 - Elegance control system servo response (C_1, C_2), for set point changes between (0.1207, 0.88) and (0.2107, 0.7226) (Jarmulak et al., 1997).	88
Figure 5-13 - Learning trial transient response for the cell mass concentration for the fittest neurocontroller. SP_2 ($C_1 = 0.2107, C_2 = 0.7226$); $SP_1 = (C_1 = 0.1207, C_2 = 0.88)$ & $SP_3(C_1 = 0.2542, C_2 = 0.5392)$. Initial condition $C_1 = 0.33$ and $C_2 = 0.36$.	91
Figure 5-14 - Learning trial transient response for the substrate conversion for the fittest neurocontroller. SP_2 ($C_1 = 0.2107, C_2 = 0.7226$); $SP_1 = (C_1 = 0.1207, C_2 = 0.88)$ & $SP_3(C_1 = 0.2542, C_2 = 0.5392)$. Initial condition $C_1 = 0.33$ and $C_2 = 0.36$.	92
Figure 5-15 - Learning trial flow rate control action for the fittest neurocontroller. SP_2 ($C_1 = 0.2107, C_2 = 0.7226$); $SP_1 = (C_1 = 0.1207, C_2 = 0.88)$ & $SP_3(C_1 = 0.2542, C_2 = 0.5392)$. Initial condition $C_1 = 0.33$ and $C_2 = 0.36$.	92
Figure 5-16 - Set point change (learned behaviour) cell mass concentration response from stable $SP_1 = (C_1 = 0.1207, C_2 = 0.88)$ to unstable $SP_3(C_1 = 0.2542, C_2 = 0.5392)$ region of the state space.	94
Figure 5-17 - Set point change (learned behaviour) substrate conversion response from stable $SP_1 = (C_1 = 0.1207, C_2 = 0.88)$ to unstable $SP_3(C_1 = 0.2542, C_2 = 0.5392)$ region of the state space.	94
Figure 5-18 - Set point change (learned behaviour) control action from stable $SP_1 = (C_1 = 0.1207, C_2 = 0.88)$ to unstable $SP_3(C_1 = 0.2542, C_2 = 0.5392)$ region of the state space.	95
Figure 5-19 - Open loop response for the cell mass concentration with $w = 0.75$ []. Initial condition $C_1 = 0.65$ and $C_2 = 0.98$. A stable steady state point is reached $C_1 = 0.1207$ [] and $C_2 = 0.88$ [].	96
Figure 5-20 - Open loop response for the substrate conversion with $w = 0.75$ []. Initial condition $C_1 = 0.65$ [] and $C_2 = 0.98$ []. A stable steady state point is reached $C_1 = 0.1207$ [] and $C_2 = 0.88$ [].	96
Figure 5-21 - Closed loop response for cell mass concentration to setpoint ($C_1 = 0.1207$ [], $C_2 = 0.88$ []) from initial condition $C_1 = 0.65$ [] and $C_2 = 0.98$ [].	97
Figure 5-22 - Closed loop response for substrate conversion to setpoint ($C_1 = 0.1207, C_2 = 0.88$) from initial condition $C_1 = 0.65$ and $C_2 = 0.98$.	97
Figure 5-23 - Closed loop control action to setpoint ($C_1 = 0.1207, C_2 = 0.88$) from initial condition $C_1 = 0.65$ and $C_2 = 0.98$.	98
Figure 5-24 - Set point change cell mass concentration response from stable ($C_1 = 0.1207, C_2 = 0.88$) to untrained unstable steady state ($C_1 = 0.1707, C_2 = 0.8032$). Cell mass concentration steady state offset $\cong 0.001$ [].	99
Figure 5-25 - Set point change substrate conversion response from stable ($C_1 = 0.1207, C_2 = 0.88$) to untrained unstable steady state ($C_1 = 0.1707, C_2 = 0.8032$). Substrate conversion steady state offset $\cong 0.002$ [].	99
Figure 5-26 - Set point change control action from stable ($C_1 = 0.1207, C_2 = 0.88$) to untrained unstable steady state ($C_1 = 0.1707, C_2 = 0.8032$).	100
Figure 5-27 - Neurocontroller response to set point changes investigated by Harris et al. (1998). Steady state offset for final setpoint change ($C_1 = 0.0785, C_2 = 0.9345$) $\cong 0.002$ [].	101
Figure 5-28 - Neurocontroller response to set point changes investigated by Harris et al. (1998). Steady state offset for final setpoint change ($C_1 = 0.0785, C_2 = 0.9345$) $\cong 0.005$ [] in substrate concentration.	101
Figure 5-29 - Manipulated variable response for set point changes proposed by Harris et al. (1998).	102
Figure 5-30 - Model $\gamma = 0.48$ for controller training. Effective γ assumed to be variable and dependent on process variables. Effective γ for each time step selected gaussian distribution around $\gamma = 0.48$ with STD = 0.01.	103
Figure 5-31 - Transient response for cell mass concentration with process / model mismatch in γ . At each time step the effective γ is assume to be within a gaussian distribution from the mean $\gamma = 0.48$, with a STD of 0.01. This corresponds to > 2% error in the model parameter as described by Brengel & Seider (1989).	103
Figure 5-32 - Transient response for substrate concentration with process / model mismatch in γ . At each time step the effective γ is assume to be within a gaussian distribution from the mean $\gamma = 0.48$, with a STD of 0.01. This corresponds to > 2% error in the model parameter as described by Brengel & Seider (1989).	104
Figure 5-33 - Manipulated variable response with process / model mismatch in γ .	104
Figure 5-34 - Model $\beta = 0.02$ for controller training. Effective β assumed to be variable and dependent	

on process variables. Effective β for each time step selected with gaussian distribution around $\beta = 0.02$ with STD = 0.004.	105
Figure 5-35 - Transient response for cell mass concentration with process / model mismatch in β . At each time step the effective β is assume to be within a gaussian distribution from the mean $\beta = 0.02$, with a STD of 0.004. This corresponds to > 20% error in the model parameter as described by Brengel & Seider (1989).	105
Figure 5-36 - Transient response for substrate concentration with process / model mismatch in β . At each time step the effective β is assume to be within a gaussian distribution from the mean $\beta = 0.02$, with a STD of 0.004. This corresponds to > 20% error in the model parameter as described by Brengel & Seider (1989).	106
Figure 5-37 - Manipulated variable control action response with process / model mismatch in β .	106
Figure 5-38 - Assumed variable substrate feed concentration with gaussian distribution around 1, with STD = 0.05.	108
Figure 5-39 - Transient response for cell mass concentration with variable substrate feed concentration. Substrate feed assumed to vary in gaussian band around 1, with a STD = 0.05.	108
Figure 5-40 - Transient response for substrate concentration with variable substrate feed concentration. Substrate feed assumed to vary in gaussian band around 1, with a STD = 0.05.	109
Figure 5-41 - Manipulated variable response to variable substrate feed concentration.	109
Figure 5-42 - Measured cell mass concentration (with noise) and actual cell mass concentration.	110
Figure 5-43 - Assumed sensor or inference noise for cell mass concentration. Gaussian noise assumed around actual cell mass concentration, with STD = 0.02. Effect on the cell mass concentration.	111
Figure 5-44 - Assumed sensor or inference noise for cell mass concentration. Gaussian noise assumed around actual cell mass concentration, with STD = 0.02. Effect on the substrate concentration.	111
Figure 5-45 - Manipulated variable response to sensor or inference noise on cell mass concentration measurement.	112
Figure 5-46 - Continuous Slurry Reactor with concentrated and dilute feed streams as manipulated variables.	115
Figure 5-47 - Multi equilibrium points at steady state.	117
Figure 5-48 - Reactor space time yield as a function of the overall inlet reactant concentration (X_0) (Matsuura & Kato, 1967).	117
Figure 5-49 - Transient response from initial conditions of $C_b = 0.1$ and $h = 40$. Solid line, no upper bounds on u_1 and u_2 ; alternate dots and dashes, upper bound of 10 on u_1 and u_2 ; dotted line, upper bound at 5 on u_1 and u_2 (Li & Biegler, 1988).	119
Figure 5-50 - Transient response to reactor start-up condition ($C_b = 0.1$, $h = 40$) (Brengel & Seider, 1989).	120
Figure 5-51 - Transient response to reactor start-up with process / model mismatch in C_{b1} of 20%, also showing the effect of mismatch compensation (Brengel & Seider, 1989).	121
Figure 5-52 - Transient level response for reactor start-up response ($C_o = 0.1$; $h_o = 40$). Solid line, $\Gamma = \text{diag}[1,1]$; dotted line, $\Gamma = \text{diag}[1,10]$; alternate dots and dashes, $\Gamma = \text{diag}[1,20]$ (Gattu & Zafiriou, 1992).	122
Figure 5-53 - Concentration response for reactor start-up response ($C_o = 0.1$; $h_o = 40$). Solid line, $\Gamma = \text{diag}[1,1]$; dotted line, $\Gamma = \text{diag}[1,10]$; alternate dots and dashes, $\Gamma = \text{diag}[1,20]$ (Gattu & Zafiriou, 1992).	122
Figure 5-54 - Transient response for the concentration of reactant B, from initial reactor condition $C_b = 1.97$ and $h = 103.43$.	125
Figure 5-55 - Transient response for the reactor level, from initial reactor condition $C_b = 1.97$ and $h = 103.43$.	126
Figure 5-56 - Concentrated feed flow rate control action from initial condition $C_b = 1.97$ and $h = 100$ to set point.	126
Figure 5-57 - Dilute feed flow rate control action from initial condition $C_b = 1.97$ and $h = 100$ to set point.	127
Figure 5-58 - Transient response for concentration B from initial condition $C_b = 0.1$ & $h = 40$.	129
Figure 5-59 - Transient response for the reactor level for initial condition $C_b = 0.1$ & $h = 40$.	130
Figure 5-60 - Concentrated feed flow rate control action resulting as a response to initial condition $C_b = 0.1$ & $h = 40$.	130

Figure 5-61 - Dilute feed flow rate control action resulting as a response to initial condition $C_b = 0.1$ & $h = 40$.	131
Figure 5-62 - Transient response for concentration B from Initial condition $C_b = 7.0$ & $h = 40$.	131
Figure 5-63 - Transient response for the reactor level for initial condition $C_b = 7.0$ & $h = 40$.	132
Figure 5-64 - Concentrated feed flow rate control action resulting as a response to initial condition $C_b = 7.0$ & $h = 40$.	132
Figure 5-65 - Dilute feed flow rate control action resulting as a response to initial condition $C_b = 7.0$ & $h = 40$.	133
Figure 5-66 - Concentrated feed disturbance with a gaussian distribution around the mean of $C_{b1} = 24$, with a standard deviation of 15.	134
Figure 5-67 - Open loop response from initial condition $h = 100$ & $C_b = 1.163$, with large disturbances in the inlet feed concentration (C_{b1}). $w_1 = 1.087$ and $w_2 = 0.9145$.	135
Figure 5-68 - Closed loop response for concentration B from initial condition $C_b = 3.31$ and $h = 68.85$.	135
Figure 5-69 - Closed loop response for Reactor level from initial condition $C_b = 3.31$ and $h = 68.85$.	136
Figure 5-70 - Concentrated feed flow rate control action to an initial condition $C_b = 3.31$ and $h = 68.85$.	136
Figure 5-71 - Dilute feed flow rate control action to an initial condition $C_b = 3.31$ and $h = 68.85$.	137
Figure 5-72 - Reversible Stirred tank reactor	137
Figure 5-73 - Equilibrium diagram for the $h = 0.16[m]$ (Economou & Morari, 1986).	140
Figure 5-74 - Reactor steady-state diagrams for nominal model and perturbations in the kinetic parameters of the plant. (n) nominal model, (a) -10% E_2 , (h) +10% E_2 , (b) +10% E_1 , (g) -10% E_1 , (c) -50% K_{01} , (e) +50% K_{01} , (d) +50% K_{02} , (f) -50% K_{02} (Patwardhan & Madhavan, 1993).	141
Figure 5-75 - Conversion transient response for the NLMPC1, NLMPC2 and Li & Biegler (1988); (Patwardhan & Madhavan, 1993).	143
Figure 5-76 - Reactor outlet temperature transient response for NLMPC1 and NLMPC2 (Patwardhan & Madhavan, 1993).	144
Figure 5-77 - Reactor level transient response for NLMPC1, NLMPC2 and Li & Biegler (1988); (Patwardhan & Madhavan, 1993).	144
Figure 5-78 - Flow rate into reactor for NLMPC1, NLMPC2 and Li & Biegler (1988); (Patwardhan & Madhavan, 1993).	145
Figure 5-79 - Reactor inlet temperature for NLMPC1, NLMPC2 and Li & Biegler (1988); (Patwardhan & Madhavan, 1993).	145
Figure 5-80 - +10% perturbation in E_1 (Patwardhan & Madhavan, 1993).	147
Figure 5-81 - Perturbation in E_1 (-10%) (Patwardhan & Madhavan, 1993).	147
Figure 5-82 - Conversion transient response from initial condition $x_{10} = 0.29$, $x_{20} = 436.98$ K and $h = 0.22$	151
Figure 5-83 - Liquid level transient response from initial condition $x_{10} = 0.29$, $x_{20} = 436.98$ K and $h = 0.22$	151
Figure 5-84 - Flow rate control action for initial condition $x_{10} = 0.29$, $x_{20} = 436.98$ K and $h = 0.22$	152
Figure 5-85 - Inlet temperature control action and reactor temperature outlet transient responses. Initial condition $x_{10} = 0.29$, $x_{20} = 436.98$ K and $h = 0.22$.	152
Figure 5-86 - Reactant conversion transient response from initial condition set 1 (section 6.6.1)	153
Figure 5-87 - Liquid level transient response from initial condition set 1 (section 6.6.1)	154
Figure 5-88 - Flow rate control action for initial condition set 1 (section 6.6.1)	154
Figure 5-89 - Inlet temperature control action and reactor outlet temperature transient response. Initial condition set 1 (section 6.6.1)	155
Figure 5-90 - Reactant conversion transient response from initial condition set 2 (section 6.6.1)	155
Figure 5-91 - Liquid level transient response from initial condition set 2 (section 6.6.1)	156
Figure 5-92 - Flow rate control action for initial condition set 2 (section 6.6.1)	156
Figure 5-93 - Inlet temperature control action and reactor outlet temperature transient response. Initial condition set 2 (section 6.6.1)	157
Figure 5-94 - Neurocontroller response to a process/model mismatch perturbation of -10% in E_1 .	160
Figure 5-95 - Inlet temperature control action and reactor outlet temperature response for a process/model mismatch perturbation of -10% in E_1 .	161
Figure 5-96 - Neurocontroller response to a process/model mismatch perturbation	

of +10% in E_1 .	161
Figure 5-97 - Inlet temperature control action and reactor outlet temperature response for a process/model mismatch perturbation of -10% in E_2 .	162
Figure 5-98 - Externally cooled CSTR with Van de Vusse reactions	163
Figure 5-99 - Educt, A, transient response from an initial condition $C_A = 1.14$ mol/L, $C_B = 0.82$ mol/L, $T_0 = 410.6$ K & $T_w = 375.74$ K.	169
Figure 5-100 - Product, B, transient response from an initial condition $C_A = 1.14$ mol/L, $C_B = 0.82$ mol/L, $T_0 = 410.6$ K & $T_w = 375.74$ K.	169
Figure 5-101 - Inlet flow rate control action for initial condition $C_A = 1.14$ mol/L, $C_B = 0.82$ mol/L, $T_0 = 410.6$ K & $T_w = 375.74$ K.	170
Figure 5-102 - Reactor system temperature transient responses from initial condition $C_A = 1.14$ mol/L, $C_B = 0.82$ mol/L, $T_0 = 410.6$ K & $T_w = 375.74$ K.	170
Figure 5-103 - External heat exchanger heat duty control action for initial condition $C_A = 1.14$ mol/L, $C_B = 0.82$ mol/L, $T_0 = 410.6$ K & $T_w = 375.74$ K.	171
Figure 5-104 - Educt, A, concentration transient response for three unlearned set point changes $SP_1 = 0.8$, $SP_2 = 0.95$ & $SP_3 = 0.7$.	172
Figure 5-105 - Product, B, concentration transient response for three unlearned set point changes $SP_1 = 0.8$, $SP_2 = 0.95$ & $SP_3 = 0.7$.	172
Figure 5-106 - Inlet flow rate control action for three unlearned set point changes $SP_1 = 0.8$, $SP_2 = 0.95$ & $SP_3 = 0.7$.	173
Figure 5-107 - Temperature transient responses for three unlearned set point changes $SP_1 = 0.8$, $SP_2 = 0.95$ & $SP_3 = 0.7$.	173
Figure 5-108 - Heat duty control action for three unlearned set point changes $SP_1 = 0.8$, $SP_2 = 0.95$ & $SP_3 = 0.7$.	174
Figure 5-109 - Inlet feed concentration C_{A0} disturbance to the Van de Vusse reactor, with gaussian distribution around the mean 5.1, with a standard deviation of 0.25.	175
Figure 5-110 - Educt, A, concentration transient response with inlet feed concentration disturbances	175
Figure 5-111 - Product, B, concentration transient response with inlet feed concentration disturbances	176
Figure 5-112 - Reactor inlet flow rate control action to inlet feed concentration disturbances.	176
Figure 5-113 - Temperature responses for the reactor system with inlet feed concentration disturbances.	177
Figure 5-114 - Heat duty control action to inlet concentration feed disturbances	177
Figure 5-115 - Educt, A, transient response to multiple model parameter gaussian variance.	178
Figure 5-116 - Product, B, transient response to multiple model parameter gaussian variance	178
Figure 5-117 - Inlet flow rate control action response to multiple model parameter gaussian variance	179
Figure 5-118 - Temperature response for the reactor system to multiple model parameter gaussian variance	179
Figure 5-119 - Heat duty control action to multiple model parameter gaussian variance.	180
Figure 6-1 - Complex batch distillation column configuration.	189
Figure 6-2 - Multi-Effect Batch Distillation system	190
Figure 6-3 - Heat integration of batch distillation columns	191
Figure 6-4 - Conventional sequencing for the separation of light and heavy impurities.	193
Figure 6-5 - Extractive batch distillation utilising a MEBAD column	194
Figure 6-6 - Batch Reactive Distillation in a MEBAD column as proposed by Mujtaba & Macchietto (1994)	196
Figure 6-7 - Cascade control for the MEBAD configuration proposed by Hasebe et al. (1995).	200
Figure 6-8 - Illustration of experimental setup	204
Figure 6-9 - Delphi Graphical User Interface utilised as Man Machine Interface.	208
Figure 6-10 - PI Control implementation for the experimental setup	210
Figure 6-11 - Neurocontrol implementation for the experimental setup.	211
Figure 6-12 - Evaluation flow diagram for the MEBAD online learning process.	212
Figure 6-13 - Premature failure criteria breakdown encapsulated in dashed lines in figure 6-12.	213
Figure 6-14 - Neurocontroller learning to control T_5 at a set point of $T_5 = 78.3$ °C.	215
Figure 6-15 - Neurocontroller learning to control T_{10} at a set point of $T_{10} = 69.0$ °C.	216
Figure 6-16 - PI Control of MEBAD column. Plant history for process variable T_5 and pump motor frequency, F_1 . Set point $T_5 = 78.3$ °C.	217
Figure 6-17 - PI Control of MEBAD column. Plant history for process variable T_{10} and pump motor	

frequency, F2. Set point $T_{10} = 69.0^{\circ}\text{C}$.	217
Figure 6-18 - Middle vessel and reflux drum level response utilising PI control.	218
Figure 6-19 - Neurocontrol of MEBAD column. Plant history for process variable T_5 and pump motor frequency, F1. Set point $T_5 = 78.3^{\circ}\text{C}$.	221
Figure 6-20 - Neurocontrol of MEBAD column. Plant history for process variable T_{10} and pump motor frequency, F2. Set point $T_{10} = 69.0^{\circ}\text{C}$.	221
Figure 6-21 - Neurocontroller middle vessel and reflux drum levels.	222
Figure 6-22 - Improvement in the sum of the absolute error from set point for process variables T_5 and T_{10} as the evolution progressed. The population statistics include the minimum error (best controller), the average error for the population & the maximum error (worst controller).	223
Figure 9-1 - Inverted Pendulum	238
Figure 9-2 - Pole angle transient response for the neurocontroller from an initial state $\theta = 6^{\circ}$ and $x = 1.2\text{m}$.	242
Figure 9-3 - Pole angular velocity transient response for neurocontroller from an initial state $\theta = 6^{\circ}$ and $x = 1.2\text{m}$.	242
Figure 9-4 - Cart position transient response for neurocontroller from an initial state $\theta = 6^{\circ}$ and $x = 1.2\text{m}$.	243
Figure 9-5 - Cart velocity transient response for neurocontroller from an initial state $\theta = 6^{\circ}$ and $x = 1.2\text{m}$.	243
Figure 9-6 - Neurocontroller manipulated variable actions resulting in the state variable response in figure 5-2 to figure 5-6.	244
Figure 9-7 - Response of linear controller developed on the linearised model, from an initial condition of $x_1 = 0.1\text{rad}$ & $x_2 = 1.2\text{m}$.	245
Figure 9-8 - The transient response for the pole angle in the nonlinear closed loop, from an initial condition of $\theta = 6^{\circ}$ and $x = 1.2\text{m}$.	246
Figure 9-9 - The transient response for the cart position in the nonlinear closed loop, from an initial condition of $\theta = 6^{\circ}$ and $x = 1.2\text{m}$.	247
Figure 9-10 - Manipulated variable action for the linear controller in a closed loop with the nonlinear plant.	247
Figure 9-11 - Linear controller transient response for the pole angle with a pole length of 0.59m in the nonlinear closed loop.	249
Figure 9-12 - Linear controller transient response for the cart position with a pole length of 0.59m in the nonlinear closed loop.	249
Figure 9-13 - Linear controller manipulated variable control action for a pole length of 0.59m .	250
Figure 9-14 - Transient response for the pole angle with a pole length 1.13m .	250
Figure 9-15 - Transient response for cart position with a pole length of 1.13m .	251
Figure 9-16 - Manipulated variable action with a pole length of 1.13m .	251
Figure 9-17 - Linear controller pole angle response to an applied force disturbance of $+30\text{N}$ after steady state has been reached	252
Figure 9-18 - Linear controller cart position response to an applied force disturbance of $+30\text{N}$ after steady state has been reached.	253
Figure 9-19 - Neurocontroller pole angle response to an applied force disturbance of $+30\text{N}$ after steady state has been reached	253
Figure 9-20 - Neurocontroller cart position response to an applied force disturbance of $+30\text{N}$ after steady state has been reached.	254
Figure 9-21 - Only Angle and Position data afforded to the neurocontroller as state inputs, resulting in a highly non-Markov state representation to the neurocontroller.	256
Figure 9-22 - Nominal full state variable feedback scenario.	257
Figure 9-23 - Temporal data obtained by means of state estimation utilising numerical differentiation.	257
Figure 9-24 - Learning in an environment with sensor noise. Sensor noise is added to the actual state value with a gaussian standard deviation. Angle with noise STD 0.6° . Angular Velocity STD $3\%/s$. Position STD 0.12 . Cart Velocity STD 0.075 .	258

1 INTRODUCTION

OBJECTIVES OF CHAPTER 1

- Motivate the need for nonlinear controller development.
 - Describe the state of overdesign in the process industry to avoid complex, though more economically viable, operating regimes resulting from nonlinearities.
 - Discuss the limitations of conventional control design methodologies as a solution to nonlinear controller design complexities.
 - Illustrate how biologically motivated neural *network* control may provide solutions for the effective control of nonlinear processes.
 - Objectives of this study.
-

1.1 MOTIVATION FOR NONLINEAR CHEMICAL PROCESS CONTROL

In spite of the extensive research related to self-tuning controllers and model reference control, there are many control problems in the chemical processing industries for which the current control design techniques are inadequate. Many of the limitations of current adaptive controllers arise in trying to control poorly modelled nonlinear systems (Ungar, 1991).

Many chemical processes are highly nonlinear. Nonlinearities may be intrinsic to the physics or chemistry of a process as in supercritical extraction, in which complex phase behaviour leads to a sensitive dependence of operation on operating conditions and control. Nonlinearities may also arise through the close coupling of simpler processes. For example, when heat integration is used to save energy, the process becomes tightly coupled, more multivariate and more difficult to control (Ungar, 1991).

Well-established technologies also present challenges. Distillation is a highly nonlinear process, and is one of the most widely studied control problems. Processes such as aluminium casting offer challenges in that no satisfactory first-principles model of the process exists and there are too many control parameters to experiment

with randomly. Different rates of cooling lead to different regimes of operation and produce aluminium with different properties (Ungar, 1991).

Extensive theoretical and experimental studies have likewise been made of both batch and continuous stirred tank reactors (CSTRs). Although such reactors may be (approximately) described by simple equations, they often exhibit complex behaviour such as multiple steady states and periodic and chaotic behaviour. Bioreactors also pose a complex control problem in that the dynamics of the system may vary significantly depending on the current process conditions. Model-based control techniques for such reactors require the determination of realistic process models and also the derivation of effective control laws, despite inaccurate process models and highly nonlinear processes (Ungar, 1991).

Although the percentage of units requiring some form of advanced control is small, these key process units as described above, may account for up to 40% of the gross revenue generated in the process industries. (Harris & Palazoglu, 1998)

1.2 OVERDESIGN TO AVOID COMPLEX NONLINEARITIES

The chemical industry in the United States may be cited for numerous instances of overdesign. Overdesign usually results as a response to compensating for model and design uncertainties, allowing for increases in capacity and to provide for margins of safe operation. Overdesign also results in order to avoid operation near or within complex operating regimes. Such operating regimes may be characterised by hysteresis and periodic or chaotic behaviour. Designs that circumvent these operating regions are often considered prudent (if these regions were known or expected to exist) as protection against unsafe or unreliable operation. Designs that constrain key parameters to avoid these regions of operation, may, however, prevent operation near steady-state economic optima. Despite the possible greater economic benefit in complex region operation, this overdesign approach prevails; as common process characteristics (table 1-1) that cause control difficulties for nonlinear and linear controllers alike, are more pronounced in more complex regions of operation. The

following section will highlight examples of such overdesign, which may be reduced through coordinated design, operation and control optimisation (Seider et al., 1990).

Brengel & Seider (1992) regard this overdesign phenomenon as an important opportunity and challenge facing design engineers, to obtain more economical designs that are flexible and controllable in regimes characterised by greater sensitivities to modelling errors (*process/model mismatches*) and changes in set points. In these regimes high performance servo-control (set point changes) and the rejection of disturbances are more difficult to achieve. Design, modelling and particularly control techniques that allow closer operation to these more complex regimes should sharply reduce the instances of overdesign in the process industries.

Table 1-1 - Common Process Control Design Complications

<i>Common Process Characteristics</i>
<ul style="list-style-type: none"> • Multivariable interactions between manipulated and controlled variables • Unmeasured state variables • Unmeasured and frequent disturbances • High-order and distributed processes • Uncertain and time-varying parameters • Constraints on manipulated and <i>state</i> variables • Deadtime on inputs and measurements • Sensor noise and inaccurate measurement

1.3 CONVENTIONAL AND NEURAL DESIGN METHODOLOGIES

1.3.1 Algorithmic and biological approaches to information processing

In real world applications numerous tasks exist for which it is difficult or virtually impossible to communicate in an algorithmic format, a logical or arithmetic sequence that will lead to the successful execution of the task. Although an effective algorithm for executing such tasks may be difficult to establish or may not exist, in several cases, the task to be performed may be specified exactly by means of a problem

statement. Many such tasks exist, for example, robust algorithmic software that allows a robot arm to "pick-up-an-object" in a dynamic (changing) *environment* is difficult to develop; yet the problem statement is simplistic (Hecht-Nielsen, 1988).

The design of controllers that operate with increasing independence from human interaction in unstructured and uncertain *environments*, is desirable. Autonomous operation is, however, dependent on the controller's ability to maintain robust performance, despite a variety of unexpected occurrences in its operating *environment*. The success of biological systems at performing numerous such complex tasks, may be regarded as a significant motivator and framework for the design of adaptation algorithms and robust learning techniques (Gupta & Rao, 1994).

Biological methods of processing information are fundamentally different from the methods used in conventional control techniques. In biological systems the plan to execute a task is formulated on the conscious level. In order to "pick-up-an-object" it is necessary to gauge the relative position of the hand to the object and also to compute direction vectors to complete the task. These computations are entirely performed at the subconscious level - the control task is not evaluated consciously in terms of muscle coordination or joint angles. Even though seemingly complex to mathematically describe such a task, biological systems are able to learn new tasks and adapt to a changing *environment* with relative ease (Gupta & Rao, 1994).

Using algorithmic control design techniques to control (for example) a robot arm, requires significant computational effort in that the angles of different robot joints need to be coordinated to produce a desired trajectory. The control law developed in this manner may fail completely should the desired task or the *environment* change (Gupta & Rao, 1994).

The procedure used for designing conventional controllers, such as PID controllers and model reference adaptive controllers, are *model based*. These design methods generally involve constructing an explicit mathematical model of the dynamic system to be controlled (Gupta & Rao, 1994).

Biological neuronal control mechanisms are *nonmodel based*. These mechanisms are successful at dealing with uncertainty and are able to interpret many degrees of freedom in executing a specific task in an unstructured *environment*. Neural control mechanisms are generally extremely complex and exact mathematical formulation is not attainable. Such complex tasks are, however, carried out without the formulation of a mathematical model of the task and the *environment*. In the execution of the task no integral, differential or other mathematical equations are solved (Gupta & Rao, 1994).

“It is our hypothesis that if the fundamental principles of neural computation used by biological control systems are understood, it seems likely that an entirely new generation of control methodologies can be developed that are more robust and intelligent, far beyond the capabilities of the present techniques based on explicit mathematical modelling.” (Gupta & Rao, 1994)

1.3.2 Conventional Control System Design

The conventional design methods for control systems involve constructing a mathematical model of the system's dynamics and the utilisation of analytical techniques (classical or modern) for the derivation of a control law. Such mathematical models comprise sets of linear/nonlinear differential equations/difference equations, which are usually derived with a degree of simplification or approximation. The modelling of physical systems for feedback control generally involves a balance between model accuracy and model simplicity (Gupta & Rao, 1994).

Should a representative mathematical model be difficult to obtain, due to uncertainty or sheer complexity, conventional techniques prove to be less useful. Also, even though an accurate model may be produced, the underlying nature of the model may make its utilisation using conventional control design difficult (Gupta & Rao, 1994).

In order to achieve satisfactory controller performance in the event of poorly understood dynamics, two approaches are typically followed. Should it be possible to

approximate the actual physical plant as a member of a class of systems (e.g. having first or second order dynamics), the use of robust controllers is generally applicable. The implementation of robust controllers is relatively simple when system stability and reasonable performance is the only concern (Gupta & Rao, 1994).

Adaptive control may also be utilised as a solution to the control for complex plants and in the presence of greater uncertainty. The controller parameters in the fixed control structure are adapted by an adaptive algorithm that ensures that the desired performance level is maintained. An adaptive control system monitors a performance index using the current dynamic *state* of the plant under control. By comparing the actual performance index to the desired performance index, the adaptation algorithm modifies the parameters of the controller to track the desired plant performance. Although techniques such as model reference adaptive control (MRAC) and automatic tuning regulators have been widely used, the application of these techniques to many realistic problems has been proved problematic (Gupta & Rao, 1994).

Within the scope of linear system theory, the theoretical developments for the design of robust linear controllers has culminated into a mature science. The use of robust linear and adaptive controllers may prove effective, should a nonlinear plant be operated in a narrow range of process conditions (locally linearized models appropriate); but frequently prove inadequate for controlling operations over a broad range of operating conditions. The design of general, practical nonlinear controllers also remains beyond the reach of the available theories. A number of conventional methods exist for developing nonlinear controllers for specific classes of nonlinear systems. These methods are system specific, with the result that a given design methodology may not be universally applied to all classes of nonlinear systems. The wide inappropriate use of multiple linear controllers or adaptive linear controllers in nonlinear *environments*, simply emphasises the need for the effective development of nonlinear controllers (Stephanopoulos & Han, 1996).

1.3.3 Neural Controller Design

Stephanopoulos & Han (1996) claim that the growth in "intelligent" control has been a reaction to the realisation that "*nonlinear control theory is not about to offer practical solutions to today's needs*". Also, evolution in the control paradigm has been fuelled by the need to deal with increasingly complex systems, and the need to accomplish increasingly demanding design requirements (closer to the economic optimum) with less precise knowledge of the plant and its *environment* (Gupta & Rao, 1994).

In order to deal with uncertain plant dynamics a controller needs to estimate or allow for the existence of unknown information during operation. Should the strategy for dealing with unmeasured *states* be effective, the controller may be considered to approach an optimal controller (Gupta & Rao, 1994).

A control system is deemed a learning control system, as opposed to an adaptive system, should a control strategy be developed based on past experience or performance. Adaptive systems differ from learning systems in that adaptive systems regard the current plant *state* as novel, whereas learning systems correlate past learned plant operating regimes with the current *state* and behave accordingly. Adaptive systems typically only adjust the parameters in a particular control structure. Learning systems may change the type and structure of controller used, or vary the parameters of the controller, after learning that the current controller is not performing satisfactorily (Gupta & Rao, 1994).

By endowing the controller with learning ability, the operating regime of the controller may be effectively increased. Learning occurs typically through the *evaluation* of examples or trials, making the explicit statement of control laws unnecessary. The control system is consequently able to compensate for a greater variety of changes in the plant and its *environmental* conditions. The learning feature precludes the existence of a fundamental plant model, as plant history may be utilised to develop empirical dynamic models for learning. Also a learning system may have the desired ability for ever-improving future performance, based on information the controller has gained in the past, through dynamic learning (Gupta & Rao, 1994).

Neural *networks*, with their ability to learn and their inherent massive parallel processing, introduce numerous opportunities for developing superior control structures for complex systems. Neural *networks* have shown great potential in the realm of multiple-input-multiple-output (*MIMO*) nonlinear control problems.

Neural *networks* are able to arbitrarily approximate nonlinear functions, which allows for the effective synthesis of nonlinear controllers (Hornik et al., 1989). A neural *network's* parallel processing capability also makes its use for multivariate control systems attractive. The ability of a neural *network*, inherent to its structure, to robustly generalise to region of the *state* space not encountered during learning or training, is of particular *value* in controller implementation to real world problems. A *neurocontroller* generally takes the form of a multilayer neural *network* with the adaptable parameters during learning defined as the adjustable *weights*. *Neurocontrollers* solve control problems by mapping *process variables* sensor readings into calculated *actions* for *manipulated variable* outputs (Gupta & Rao, 1994).

1.4 REINFORCEMENT LEARNING FOR NEURAL CONTROLLER DESIGN

The idea that humans learn by interacting with their *environment* is fundamental to the nature and mechanism of human learning. A newborn infant has no explicit teacher, but the infant does have a direct sensorimotor connection to its *environment*. Interacting with its *environment* produces a wealth of information regarding cause and effect, regarding the consequences of *actions*, and also regarding which behavioural pattern will lead to achieving a specific goal. Knowledge of cause and effect patterns in an *environment* provide the learner with the ability to influence the *environment* through a particular pattern of behaviour (Sutton & Barto, 1998).

Reinforcement learning is defined as a learning problem in which a learner or *agent* (*reinforcement learning agent*) must learn a particular behaviour through trial-and-error with a dynamic *environment*. An *agent* is initially unfamiliar with which *actions* to execute to successfully complete a given task or achieve a *reward*. This requires

the *agent* to search for an appropriate set of *actions* which yield the highest possible *reward*. The solution to the *reinforcement learning* problem involves mapping certain *environmental states* to *actions* through a computational approach. Its potential is beguiling - a means to program *agents* using cause and effect (*reward* and punishment) interaction information, without needing to specify how the goal is to be reached (Kaelbling et al., 1996).

1.5 OBJECTIVES OF THIS STUDY

As *reinforcement learning* has generally found application in areas other than process control (for example, robotics and game theory), the application of this control *policy* learning technique is investigated for evolving robust *neurocontrollers* for process control application. Four process control simulation benchmarks are considered, which exhibit highly nonlinear behaviour and consequently complex regions of operation. These simulation studies include developing optimal *neurocontrollers* for various types of reactors - a bioreactor, a catalytic reactor, a nonisothermal reactor and a Van de Vusse reactor. Also, the analysis is extended to an investigation for learning a optimal control *policy* for a real world novel batch distillation pilot plant. A novel *evolutionary reinforcement learning* algorithm *SANE* (Symbiotic, Adaptive, Neuro Evolution) is implemented for developing the desired optimal *neurocontrollers*.

The objectives of this study are as follows -

- Demonstrate the applicability of *evolutionary reinforcement learning* techniques, in particular the *SANE* algorithm, for providing complex simulation process control problems. Evaluate the ability of the developed *neurocontrollers* to provide robust control performance over a wide range of operating conditions, process disturbances and plant model uncertainties. This *evaluation* provides a direct means of determining the degree of *generalisation* afforded by the neural *network* control structure.

- Provide performance and robustness comparisons between previously considered control strategies (model predictive control) and the *neurocontroller* control strategies developed with *SANE*.
- Conventional approaches to optimal nonlinear process design and process control development, include a two step calculation process. Firstly, the most economic region of operation must be found utilising typically nonlinear programming techniques. Once the region of greatest economic benefit has been established, advanced process control techniques, such as model predictive control, may be used to develop controllers that are robust in this operating region. This study aims to show that *SANE* is an effective tool for integrating the process design and process control development tasks into a single calculation step.
- Construct a pilot plant novel batch distillation column. Investigate the applicability of the *SANE* learning algorithm for online control *policy* determination (learning) for this batch column.

To achieve the above objectives, Delphi 4.0 was utilized to create an event driven implementation of the *SANE* algorithm for learning from dynamic systems. To implement the *SANE* algorithm on the constructed batch distillation column, a man-machine interface was programmed for signal processing of the plant instrumentation.

2 REINFORCEMENT LEARNING

OBJECTIVES OF CHAPTER 2

- Highlight complexities that need to be addressed for effective *reinforcement learning* in a dynamic *environment*. These elements of *reinforcement learning* are discussed in the preceding sections.
 - *A glossary is provided as appendix C for reading with this chapter.*
-

2.1 ELEMENTS OF REINFORCEMENT LEARNING

Reinforcement learning is a computational approach to understanding and automating goal-directed learning and decision-making, as introduced in section 1.4. It is distinguished from other computational approaches by its emphasis on learning from direct interaction with its *environment*, without relying on exemplary supervision or complete models of the *environment*. A learning *agent* consequently needs to be able to sense the *state* of its *environment* and take *actions* which affect the current *environmental state*. Furthermore, the *agent* requires a clearly defined goal relating to a desired *environmental state* or what constitutes the successful completion of a task. *Reinforcement learning* thus provides a framework for explicitly defining the interaction between a learning *agent* and its *environment* in terms of *states*, *actions* and *rewards* (Sutton & Barto, 1998).

There are two main strategies for solving *reinforcement learning* problems. The first is to search in a space of possible behaviour patterns in order to find a complete behavioural pattern that encompasses effective performance in the *environment*. This approach has been taken utilising *genetic algorithms* and genetic programming, as well as other novel search techniques, such as simulated annealing and tabu search. The second is to use statistical techniques and dynamic programming methods to

estimate the benefit of taking *actions* in particular *states* of the *environment*. Such statistical approaches include *temporal difference* methods, of which the *Q-learning* algorithm has enjoyed the greatest amount of application (Kaelbling et al., 1996).

2.2 REINFORCEMENT LEARNING VS. SUPERVISED LEARNING

As knowledge acquisition in the design of expert systems has proven to be a bottleneck, research in machine learning has attempted to compensate by automating the knowledge acquisition process and by broadening the base of accessible sources of knowledge (Grefenstette et al., 1990).

The selection of an appropriate learning technique is dictated by the nature of the task to be performed and the format of available knowledge. Should the task be classification, along with the availability of a large training set, supervised learning techniques may be utilised (Grefenstette et al., 1990).

For artificial neural *networks* (*ANN*), learning is typically accomplished using examples of correct *actions* or behaviour. Supervised learning is based on direct comparison between the actual output of an *ANN* and the desired correct output (target output). It is often formulated as the minimisation of an error function such as the total mean square error between the actual output and the desired output, summed over all available data. A gradient descent-based optimisation algorithm such as backpropagation may be used to adjust connection *weights* in the *ANN* in an iterative fashion in order to minimise the error (Yao, 1999).

Many real world practical problems that may be subject to automated learning, do not fit into the supervised learning model. A class of problems described as sequential decision tasks, for which effective performance *evaluation* is only possible after a series of different *actions* have been executed, often do not comply with the requirements that allow for the use of supervised (traditional machine learning) techniques. The necessary *environmental* (*domain*) information for generating the target output is either unavailable or too costly to obtain. In this case, the development of effective decision tasks is subject to testing the developed controller against a simulated model of the task *environment*. The controller may subsequently be

incrementally modified based on the simulation experience (Greffenstette et al., 1990).

A decision making *agent* interacts with a discrete dynamic system in an iterative fashion. Initially the system may be at some *state*. The *agent* observes the current *state* of its *environment* (or a representation thereof) and selects a control *action*. The discrete dynamic system enters a new *state* and a payoff is assigned. The objective is to determine a sequence of tasks that will maximise the expected total payoff. The input signals to the *agent* only provide a general measure of proficiency in the task. These input signals do not explicitly direct the *agent* towards a control *policy*. Learning is based only on the information of whether or not the actual output is correct. The design of sequential decision *agents* by this approach involves a rapidly changing *state environment*, which makes this type of *agent* best suited for reactive rather than projective planning (Greffenstette et al., 1990).

The motivation for using simulation models for learning control policies, lies in that making mistakes on real systems may have costly or dangerous consequences. Learning may require experimenting with strategy policies that may occasionally produce unacceptable results in a real world *environment*. The use of simulation models have been instrumental in several machine learning efforts (Greffenstette et al., 1990).

2.3 THE AGENT-ENVIRONMENT INTERFACE

In the standard reinforcement-learning model (figure 2-1), an *agent* is connected to its *environment* via perception and *actions*. At each interaction step the *agent* receives an input which is either a full or partial *state* representation of the *environment*. The *agent* chooses an *action*, based on the input *state* information to generate an output. The *action* changes the *state* of the *environment*, the *value* of this single (discrete) *state* transition is communicated to the *agent* through a scalar reinforcement signal. The *agent's* behaviour *policy* should choose *actions* that tend to increase the long-run sum of single *state values* of the reinforcement signal. It can learn to do this over time

by systematic trial and error, guided by a wide variety of algorithms (Kaelbling et al., 1996).

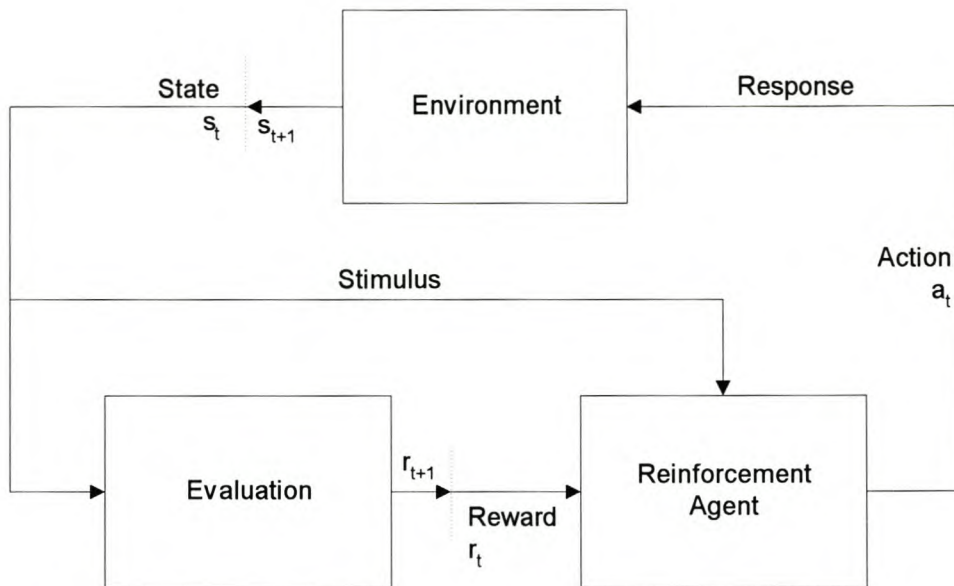


Figure 2-1 - Agent-Environment Interaction

The *agent's* task is to find a *policy*, π , mapping *states* to *actions*, that maximises some long-run measure of reinforcement. The *environment* will generally be non-deterministic, that is, taking the same *action* in the same *state* on two different occasions may result in different next *states* and/or different reinforcement *values*. Also, it is generally assumed that the *environment* is stationary; that is, that the probabilities of making *state* transitions or receiving specific reinforcement signals do not change over time. Many algorithms, however, are effective in slowly-varying non-stationary *environments*. It is necessary for the *agent* to gather useful experience regarding the possible system *states*, *actions*, transitions and *rewards* on an active basis to perform optimally (Kaelbling et al., 1996).

The *agent's actions* not only determine its immediate *reward*, but also probabilistically the next *state* of the *environment*. The *agent* must take into account the *value* of the next *state* in completing the task, along with the *states* immediate *reward* when it decides which *action* to select. The model of long-run optimality the *agent* is using determines exactly how it should take the *value* of the future into account. The *agent* will have to be capable of learning from delayed reinforcement. It

may take a long sequence of *actions*, receiving insignificant reinforcement to finally arrive at a *state* with high reinforcement. The *agent* must be able to learn which of its *actions* are desirable based on *reward* that is likely to take place arbitrarily far into the future (Kaelbling et al., 1996).

For a *Markov* decision process (MDP) the effects of *actions* (i.e. the next *state* and the immediate *reward* generated) depend only upon the current *state*. For Non-*Markov* decision processes the transition function may need to map a past and present *state* to a particular *action* (Whitehead & Lin, 1995).

A *Markov* decision process is described by the tuple (S, A, T, R) where S is the set of possible *states*, A is the set of possible *actions*, T is the *state* transition function, and R is the *reward* function (figure 2-1). At each instance the *environment* occupies exactly one *state* from S , and accepts one *action* from A . S and A are assumed to be discrete and finite. *State* transitions are modelled by the transition function, T , which specifies the next *state* of the *environment* as a function of the current *state* and the *agent's action* ($T(S_t, A_t) \rightarrow S_{t+1}$). *Reward* generated by the *reward (evaluation)* function specifies the expected instantaneous scalar *reward* as a function of the current *state* and *action*. ($R(S_t, A_t) \rightarrow R_{t+1}$) (Whitehead & Lin, 1995).

The optimum *reward temporal difference* of a *state* is the expected discounted sum of *rewards* that the *agent* will gain should it start in a particular *state* and executes the optimal *policy* (Kaelbling et al., 1996).

2.4 MODELS OF OPTIMAL BEHAVIOUR

To provide effective reinforcement to an *agent*, a specification is required regarding how the *agent* should take the future into consideration in *actions* that it is taking in the present. Three models of optimal behaviour are generally considered.

The *finite horizon model* requires the *agent* to optimise its expected *reward* for the next h steps.

$$R\left(\sum_{t=0}^h r_t\right) \quad (2-1)$$

No consideration is given to the time period after h time steps has elapsed. The scalar, r_t , is the *reward* received t steps into the future. This model may be used in two ways. In the first, the *agent* has a non-stationary *policy*, that is, one that changes over time. Its first *action* is termed a h -step optimal *action*. This is the best *action* available given that h steps remain in which to act and gain *reward*. On the next step a $(h-1)$ -step optimal *action* is taken, until it finally takes a 1-step optimal *action* and terminates. The second implementation involves receding-horizon control, in which the h -step optimal action is always taken. The *agent* always acts according to the same *policy*, but the *temporal difference* of h limits how far it looks into the future when choosing its actions. The finite-horizon model is not always appropriate, as the precise length of the *agent's* life may not always be known in advance (Kaelbling et al., 1996).

The *infinite-horizon discounted model* takes the long-run *reward* of the *agent* into account, but *rewards* in the future are discounted according to the discount factor γ , where $0 \leq \gamma \leq 1$.

$$R\left(\sum_{t=0}^{\infty} \gamma^t \cdot r_t\right) \quad (2-2)$$

In this model the *agent* is required to show a performance increase from one time step $(t-1)$ to another (t) , in order to receive the same *reward* as in the previous time step $(t-1)$. Should the *temporal difference* of γ be 0.91, a 10% increase in performance is required from one time step to another, to maintain the same *reward* as for the previous time step $(k-1)$.

The *average-reward model* requires the *agent* to take actions that optimise its long-run average *reward*.

$$\lim_{h \rightarrow \infty} R \left(\frac{1}{h} \cdot \sum_{t=0}^h r_t \right) \quad (2-3)$$

This *policy* is referred to as a *gain optimal policy*, as it may be regarded as a limited case of the infinite-horizon discounted model, as the discount factor approaches 1. A difficulty that arises with this criterion, is that no means exist to distinguish between two policies with the same average *reward*. No distinction is made between a *policy* that gains much *reward* in the initial phase, as opposed to a *policy* that gains much *reward* in the latter phase of its execution. Also, *reward* gained on any initial prefix of the *agent's* life is overshadowed by the long-run average performance.

2.5 CREDIT ASSIGNMENT

Control problems that rely exclusively on the uninformative (comparatively) failure signal for feedback, present a challenging problem to *reinforcement learning* techniques. Consider a sequence of discrete (binary 1 or 0) control actions (or responses) followed by a failure signal (F). At any instance the *agent* may thus select only between a high (1) or a low (0) output signal in response to its *environment*. In the following action sequence 100011110000F a classic *credit assignment* problem exists (Whitley et al., 1993).

In such an action sequence, the problem exists as to how to distribute credit for success or failure among the many decisions that may have been involved in producing success or failure. Each ultimate success (or failure) is thus associated with a vast number of internal decisions. For learning to occur from such sparse reinforcement information, the task must be divided into components. The measure of success lies in the completion of the entire goal. If a goal is achieved, its subgoals (rather the *transformation* function that produced the subgoal) are reinforced; if not such *transformations* are inhibited. Assigning $1/12$ of the credit to each *individual* task for either success or failure, would only be feasible should each task have a sufficient degree of independence. Assigning reinforcement to any real world *individual* subgoal, would thus typically involve determining the relative contribution of interrelated subgoals (Minski, 1961).

The above illustration illustrates the difficulty associated with allocating *reward* to *individual* actions, as in *temporal difference (Q-learning)* methods. In some cases *individual* input *state* sets may be misleading (e.g., when *states* are misperceived), resulting in a misallocation of appropriate *reward*. This could divert the *temporal difference* method to suboptimal regions of the solution space. In *evolutionary reinforcement learning* *reward* is normally associated with a whole sequence of *states* and decisions, not with each *individual* decision. *Credit assignment* for each *individual* decision is made implicitly, as poor *agents* generally select poor *individual* decisions. *Rewarding* a control *policy* based on a sequence of *states* provides implicit learning robustness, as opposed to *rewarding individual states* (which could be misperceived) separately.

2.6 THE MARKOV ASSUMPTION

2.6.1 The Markov Property

The greater majority of learning algorithms based upon *reinforcement learning*, focus on decision tasks that are described as *Markov* decision processes. A *Markov* control task implies that at each point in time the *agent* -

- Directly observes the full *state* of the *environment*.
- The effects of the action, depend only upon the action taken and the current *state*.

The assumption that the *agent*, at each instance in time, has access to all *state* information (either through external sensors or internal representation) that is relevant to predicting the outcome of a given action, is known as the *Markov* assumption. In a *Markov* decision process (MDP), the effects of actions (i.e. the next *state* and the immediate *reward* generated) depend only upon the current *state*. Process models of this type are said to be memoryless and to satisfy the *Markov* property. The *Markov* property implies that knowledge of the current *state* is always sufficient for optimal control. Thus, even though it may be possible to devise action-selection strategies whose decisions depend upon additional information (e.g. a history trace), these

strategies cannot outperform the best decision *policy* that depend only upon knowledge of the current *state*. Formally, a decision task is non-*Markov* if information above and beyond knowledge of the current *state* can be used to better predict the dynamics of the process and improve control (Whitehead & Lin, 1995).

Although the *Markov* assumption holds for a wide variety of control problems, there are numerous tasks that are inherently non-*Markov*. These tasks are referred to as *hidden state tasks*, as information that pertains to describing the dynamic *state* of the *environment* is hidden from the *agent's* representation of the current *state*. Two significant control problems that are not naturally formulated as *Markov* decision processes, are -

- Where the system has a significant degree of control over the information collected by its sensors (e.g. in active vision).
- Where the system has a limited set of sensors that are not able to provide adequate information regarding the present *state* of the *environment*. (Whitehead & Lin, 1995)

2.6.2 The ubiquity of non-*Markov* tasks

Markov tasks are an ideal. Non-*Markov* tasks are the norm. An *agent* that can be uncertain about the *state* of the external task necessarily faces an internal decision problem that is non-*Markov*. Sources of uncertainty abound. Sensors have physical limitations and are rarely perfectly matched for the task. Sensor data is typically noisy, unreliable, and full of spurious information. Sensors also have limited range. Information may also be hidden in time, that is, the *agent* requires past and current *state* information in order to take appropriate action (Whitehead & Lin, 1995).

Such hidden *state* tasks thus arise when temporal features (such as velocity and acceleration) are important for optimal control, but not included in the system's primitive sensor set. Even if perfect sensors were available, many control problems are too ambiguous or ill-posed to specify a *state* space in advance. Indeed, part of the *agent's* task may be to discover a useful *state* space for the problem. Integrating the learning and active perception tasks invariably leads to non-*Markov* decision tasks.

Active perception requires an intelligent *agent* to actively control its sensors in order to sense and represent information that is exclusively relevant to its immediate ongoing activity. If an *agent* must, as part of the task, learn to control or select appropriate sensors, its internal control problem will necessarily be non-*Markov*. This realisation is apparent in that during learning there will be periods of time when the *agent* will improperly control or utilise the available sensors, fail to attend to a relevant piece of information, and fail to unambiguously identify the *state* of the external task (Whitehead & Lin, 1995).

2.6.3 Difficulties for reinforcement learning

Intelligent control systems must, however, deal with information limitations posed by their sensors. The straightforward application of traditional *reinforcement learning* methods to non-*Markov* decision problems in many cases yields sub-optimal results and in some cases severely degraded performance. These difficulties stem from the *agent's* inability to obtain accurate estimates of the *value* of the current actual (as opposed to perceived) *state* and therefore can not accurately gauge the applicability of the decision *policy*. Also, relevant *states* (hidden *states*) may be omitted as sensory input to the *agent*. Due to poor sensor information two actual *states* in the *environment* may be misperceived as a single state in the *agent's* internal representation (perceptual aliasing) (Whitehead & Lin, 1995).

Q-learning (temporal difference method), for example, cannot be guaranteed to converge in non-*Markov environments*. Small breaches of the *Markov* requirement are well-handled by *Q-learning*, but simple *environments* have been constructed that cause *Q-learning* to oscillate (Kaelbling et al., 1996). *Evolutionary* methods have advantages on problems in which the learning *agent* cannot accurately sense the state of its *environment* (Sutton & Barto, 1998).

2.7 EXPLORATION AND EXPLOITATION

In *reinforcement learning* a balance must be maintained between *exploration* and *exploitation* in the *agent's* search for an effective solution. In order to gain a great deal of *reward*, an *agent* must prefer actions that it has tried in the past and found to be effective in producing *rewards*. However, in order to discover such actions, it needs to perform actions that it has not attempted in the past. The *agent* has to exploit what it has learned in order to obtain *reward*, but it also has to explore to pursue more effective actions in the future. Neither *exploration* nor *exploitation* may be pursued exclusively, without failing at the task. The *agent* must try a variety of actions and progressively favour those that produce improved *rewards* (Sutton & Barto, 1998).

Simple ad-hoc techniques illustrate the demands in balancing *exploration* and *exploitation*. These techniques are rarely the best choice for models of optimality, but serve as reasonable heuristics to the problem of balancing *exploration* and *exploitation* in small *state space domains* (Kaelbling et al., 1996).

A *greedy strategy* entails always choosing the action with the highest estimated *reward*. This approach is flawed when sampling error biases cause the *reward value* of the best action to be less than the *reward* obtained from a suboptimal action. The suboptimal action is consequently always selected, leaving the true optimal action starved of data and its superiority thus remains undiscovered. An *agent* must explore to prevent this type of outcome. A useful heuristic is *optimism in the face of uncertainty*, in which actions are selected greedily, but strongly optimistic prior beliefs are attached to their intrinsic *value* despite the actual assigned *reward*, requiring strong negative evidence to eliminate the action from future consideration. The risk of starving an optimal but poorly sampled action remains, but this type of approach allows for arbitrary risk management. A wide range of such techniques exist for application, include the interval *exploration* method, the *exploration* bonus, curiosity-driven *exploration* and, the *exploration* mechanism in prioritised sweeping (Kaelbling et al., 1996).

Randomised exploration strategies always take the action with the best estimated expected *reward*, but with a probability p select an action at random. Such a strategy may begin with a large *value* for p to encourage initial *exploration*, with the *value* of p slowly decreasing as the search progresses. The randomly selected action is, however, a non-directed choice, in that this strategy will selected between a promising alternative and a clearly flawed alternative with equal probability (Kaelbling et al., 1996).

Exploration may be made more efficient should it be based on second-order information about the certainty or variance of the estimated *values* of actions. An interval estimation algorithm stores statistics for each action; the number of successes and the number of trials. A confidence interval is calculated on the success probability of each action. The action with the highest level of confidence is selected. The lower the confidence interval for a particular action, the greater the probability of *exploration*.

In *environments* with many possible *states*, in which reinforcement is awarded immediately after the action has been executed (no time delay in action's effect), the above strategies may be replicated once for each possible *state*. Should the size of the *state* space make *generalisation* imperative, these techniques need to be integrated with *generalisation* tools (such as neural *networks*).

Many of these techniques are directed to converge to a *policy* from which *exploration* is rarely undertaken. This may be appropriate for stationary *environments*, barring that such *convergence* has not found a local optimum solution. In non-stationary *environments*, *exploration* must continue to take place in order to perceive changes in the *environment*, making the use of converging techniques inappropriate (Kaelbling et al., 1996).

Evolutionary reinforcement learning methods effectively balance *exploration* and *exploitation* of the solution space by maintaining a *population* of different strategies. As multiple decision policies are represented, *evolutionary* techniques sample various *policy* decisions. *Evolutionary* search methods progress non-randomly by probabilistically assigning a greater number of *evaluations* to *individuals* that display

more effective behaviour than an average *individual*. This may be regarded as an *exploitation* step. The *evolutionary operators* (i.e., *crossover*) produce *offspring* that represent an *exploration* step into the solution space. Other than in the case of random *exploration* strategies, this *exploration* step is directed towards solutions that have a greater probability for producing greater *rewards*. This directed *exploration* step results as *offspring* of effective *parents* will probabilistically have a similar or improved performance in comparison with their *parents*. *Exploitation* is also maintained as *parents* with high *fitness values* are often transferred unchanged to the next generation for *evaluation*. Maintaining genetic *diversity* ensures that the learning algorithm is also able to adapt to non-stationary *environments*.

2.8 GENERALISATION

As *reinforcement learning* problem definitions become more complex, the number of possible *states* for the *environment* generally grows exponentially with the complexity of the task. In large *state* spaces, *agents* cannot observe every possible *state* and must apply action decisions learned in observed *states* to unobserved *states*.

Real world applications preclude the storing of the *values* of all *states* and actions. Except in very small *environments*, such storage of *value* representations presents an impractical memory requirement. It also makes inefficient use of experience. In a large, smooth *state* space it is generally expected that similar *states* will have similar *values* and similar optimal actions. A more compact representation than a simple look-up table is thus required. The difficulty in learning in large *state* spaces is addressed through *generalisation* techniques, which allow for compact storage of learned information and sharing of knowledge between "similar" *states* and actions. The reinforcement-learning model requires the storage of a variety of mappings; including *state* to action *policy* transitions ($S \rightarrow A$) and deterministic *state* transitions ($S_t \times A \rightarrow S_{t+1}$). *Generalisation* tools such as artificial neural *networks*, fuzzy logic and genetic programming may be utilised to compactly represent these mappings (Kaelbling et al., 1996).

2.9 CONTROL POLICY REVISIONS AND MEMORY

Learning from interaction with the *environment* implies that the learning *agent* needs to change its current decision *policy* to a decision *policy* that has produced greater *reward*. Consideration needs to be given to as to how much greater the *reward* must be, before it is considered prudent to update the current control *policy* (Moriarty, 1997).

For neural *network agents* the updating of the decision *policy* entails changing the *weights* and connections of the associated *neurons*. Should only a single *agent* be evaluated in the *environment* (as for *temporal difference* methods), updating the control *policy* could be detrimental to future performance should the *reward* information presented be inaccurate or misleading. Sensor reading errors may present a source of such inaccurate *state* representations. Also, disturbances in the *environment* may randomly cause the plant to be in a region of the *state* space with high associated *reward*. In other words, the *agent's* actions were not responsible for the favourable *state* of the *environment*, but rather some random occurrence in the *environment*. *Policy* updates based on erroneous information, may cause the *agent's policy* to drift from the optimal strategy.

Despite the aforementioned, *temporal difference* approaches typically revise the single decision *policy* after each action. The revisions are small, as the revisions are only based on a single action's cause and effect. Frequent *policy* updates based on local information may, however, adversely affect *generalisation*. Updates based on a single *state* and decision pair in one instance may cause *state* and decision pairs in other instances to be under or overestimated. This results as updates to neural *network weights* affect *policy* decisions for many other *states* (Moriarty, 1997).

Evolutionary strategies make revisions by applying genetic *operators* to the *population*, only once several *agents* in the *population*, each over a sequence of actions, have been evaluated. The revisions are typically larger, as the revision is based on a wider pool of more global information. Updating with less frequency using information gathered from several *states* and actions, reduces the probability of error

that may result due to reward biasing from occurrences in the *environment* (Moriarty, 1997).

An attractive feature in maintaining a *population* of *agents* (policies) lies in that even though a single *agent's* reward may not reflect the *agent's* intrinsic *value*, *evolutionary* algorithms tend to maintain a degree of robustness. A *genetic algorithm's* replacement *policy* typically replaces the weakest *individuals* in the *population*. Provided that the *agent's* biased reward (*fitness*) is not significantly lower than its actual intrinsic *reward*, the *agent* may survive into the next generation without change. Another *evaluation* opportunity is thus afforded for *evaluation*.

2.10 CONCLUDING REMARKS

Reinforcement learning presents a means by which effective control policies may be derived from direct interaction with a dynamic *environment*. The methodology is attractive as control policies may be derived based on sparse information regarding the *environment*, in which a particular behaviour must be learned. The ability of reinforcement *agents* to generalise from few scattered facts presents an attractive means of acquiring knowledge for control purposes. All *reinforcement learning* algorithms need to address the elements highlighted in this chapter in order to effectively derive control policies from the particular *environment*. There are many different approaches through which these fundamental issues may be addressed. *Evolutionary* algorithm strategies generally provide a more effective means of dealing with these fundamental challenges. For example, *evolutionary* search strategies offer a more robust *credit assignment* strategy and impart excellent *generalisation* when evolving neural *network* control structures.

2.11 SYMBOLS FOR CHAPTER 2

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
a, A	<i>Reinforcement learning agent</i> control actions	-
h	Number of time units in a reinforce learning <i>agent's</i> life	-
r, R	<i>Reward</i> allotted to <i>reinforcement learning agent</i>	-
s, S	<i>State</i> of the <i>environment</i>	-
T	<i>State-action transformation</i> function	-

Greek symbols

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
γ	Discount factor for infinite-horizon discounted model	-

Subscript

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
t	Time	-

3 EVOLVING NEURAL NETWORKS FOR CONTROL

OBJECTIVES OF CHAPTER 3

- Describe the search mechanism principle utilised in *genetic algorithms*.
- Discuss the difficulties in developing neurocontrol policies as a result of the *genetic algorithm* search mechanism.
- Illustrate the importance of maintaining *population diversity* for complex control *policy* development. Discuss techniques proposed in literature for maintaining *diversity* in unconverged *populations* to facilitate effective search.
- Show the superiority of *implicit fitness sharing* for maintaining *diversity* over explicit techniques.
- Describe the algorithm utilised in this study, *SANE*, for control *policy* development.
- The suitability of *SANE* for evolving neural *networks*, with particular reference to how it overcomes the difficulties of evolving neural *networks* with *genetic algorithms*.
- Describe past applications of *SANE*.
- Discuss *reinforcement learning* in a process control context.
- *A glossary is provided as appendix C for reading with this chapter.*

3.1 FOUNDATIONS OF GENETIC ALGORITHMS

Genetic algorithms typically initialise by randomly generating a *population* of *encodings* (strings) that represent possible solutions to a particular problem. The "strings", or *genotypes*, are evaluated to obtain a quantitative measure of how well the "strings" are performing as possible solutions. Should possible solutions be encoded as 16 bit binary strings, the total number of possible solutions (*hyperspace*) that exist to the problem is 2^{16} . Consider the following binary string: 1101 0011 0010 1101. The string may represent, for example, a simple neural *network* with four links, where each connection *weight* is represented by four bits. The goal of genetic *recombination* is to find a set of parameters that yield an optimal or near optimal solution to the

problem. *Recombination* requires two *parents*. Using two "break-point" *recombination* a second binary string yxyy xyxx yyyx yxxy (where x and y represent 0 and 1 respectively) may be recombined with 1101 0011 0010 1101 to produce the following *offspring* -

1101 0yxx yyyx y101
yxyy x011 0010 1xxy

Reproduction opportunities are allocated such that the best strings receive more opportunities to reproduce than those which have poor performance. This bias need not be great to produce the required selective pressure to allow "artificial selection" to occur. More trials are consequently allocated to strings containing pertinent fragments of solution information (regions in the *hyperspace*) that tend to contain above-average solutions (Whitley et al. ,1990).

To understand how *recombination* on binary strings can be related to *hyperspace*, consider a "*genotype*" that is encoded with just 3 bits. With three bits the resulting search space is three dimensional and can be represented by a simple cube. Let points 000, 001, 010 and 011 be the front face of the cube. The front plane of the cube can be characterised as all the *genotypes* beginning with 0. If * is used as a wild card match symbol, then this plane can also be represented by the similarity template 0**. These similarity templates are referred to as *schemata*; each *schemata* corresponds to a hyperplane in the search space. All bit strings that match a particular schema lie in its hyperplane. In general, every binary *encoding* corresponds to a corner in a L-dimensional hypercube and is a member of $2^L - 1$ different *hyperplanes*, where L is the length of the binary *encoding*. For example, the string 011 not only samples its own corner in the hyperplane (011) but also the *hyperplanes* represented by the *schemata* 0**, *1*, 01*, 0*1, *11 and *** (Whitley et al. ,1990).

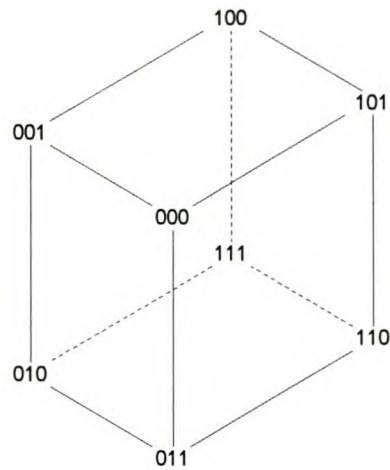


Figure 3-1 - Hyperplane Cube

The characterisation of the search space as a hyperplane is not simply a way of describing the space. It relates directly to the theoretical foundations of genetic search. Each schema (chromosome) represents a different genetic "fragment", a different combination of "alleles" (genes). When *recombination* occurs, strings are exchanging hyperplane information. For example, if 10101100 and 11011110 are recombined, the *offspring* reside in the hyperplane $1***11*0$. This represents an "order-4" hyperplane, as four bits are specified in the schema. This hyperplane contains $6 \frac{1}{4} \%$ of the all strings in the search space - all strings with 1 as first bit, 1 as fourth bit, 1 as fifth bit and 0 as eighth bit. The section of the search space that is in contention between the two strings, is -010--0- and -101--1-. That is, the bit positions represented by a dash (-) may be either 0 or 1, depending on the result of the applied genetic *operator* (such as *crossover*) (Whitley et al. ,1990).

The *offspring* resample (re-evaluate) those *hyperplanes* (*gene* combinations) inherited unchanged from the *parents*, but they resample (re-evaluate) the *gene* combinations in a new context - from a new corner in the hypercube with a different *evaluation*. By testing one new *genotype*, additional information is gained about the $2^L - 1$ *hyperplanes* that intersect at a corner in the hypercube where that *genotype* resides (Whitley et al. ,1990).

After a *genotype* has been tested, it is probabilistically given the chance to reproduce at a rate that reflects its "*fitness*" relative to the remainder of the *population*.

Recombining the "genetic material" from two *parents* by crossing the binary *encodings* allows other *hyperplanes* in the search space to be sampled (resampled), but these new probes will be biased towards those *hyperplanes* that have displayed above-average performance. If a schema is most often found in *genotypes* that have above average performance, this indicates that the schema may represent a hyperplane that on average contributes to optimising the target problem with greater effectiveness than the average *genotype* in the *population*. This hyperplane thus represents a section of the search space that requires greater *exploration*. Recombining allows for *hyperplanes*, which have displayed an above average performance, to increase their representation in the *population*. Genetic search thus proceeds by changing the sampling rate of *hyperplanes* in the *population* (Whitley et al. ,1990).

Genetic algorithms are capable of performing a global search of a solution space as there is reliance on hyperplane sampling to guide the search, instead of searching along the gradient of a function as with other optimisation routines. The property of genetic search to efficiently samples numerous *hyperplanes* in parallel is referred to as *implicit parallelism*. This implicit parallelism property allows for a robust search method capable of effective directed search without using gradient information (Whitley et al., 1993).

3.2 DIFFICULTIES IN EVOLVING NEURAL NETWORKS

Genetic algorithms inherently do not scale up effectively. A bias exists against *schemata* that are highly separated in the binary *encoding*. As the string length increases, a greater proportion of the sampled *hyperplanes* will span more than half the string, which will thus have a greater probability of being disrupted by *crossover*. This observation is assumed to contribute to scale-up (longer binary *encodings* may be required to represent more complex solutions) problems with binary *encodings* (Whitley et al. ,1990).

One difficulty in evolving neural *networks* with *genetic algorithms* is that multiple symmetric representations exist for any single neural *network*. Recombining two functionally effective, but structurally dissimilar *networks*, may result in *offspring* that

are not viable solutions or present a considerable loss in functionality. Such inconsistent performance feedback to the *genetic algorithm*, present as high variance hyperplane sampling (section 3.1), may significantly retard the genetic search. For example, assume (figure 3-2) that four hidden nodes perform tasks A, B, C and D as a solution to some problem. With the same connectivity pattern, the two *network* in figure 3-2 are functionally identical. The position of a functional hidden node in the feed-forward neural *network* has no effect on the neural *networks* performance. Recombining these two functionally similar neural *networks*, could result in *children* with functionality ABDB and CACD (one point *crossover* on a hidden node boundary). These two combinations of functionality do not represent a solution to the problem, as one functional hidden node is absent from each *offspring network*. The probability of different structural/functional mappings increases as the number of hidden nodes increase, should the hidden nodes have identical connection patterns. Also, functionally similar hidden nodes may have sets of *weights* with a different scaling. Recombining *networks* with dissimilar *weights* may result in *offspring* with poor performance. This *network recombination* problem is referred to as the structural/functional mapping problem (Whitley et al. ,1990).

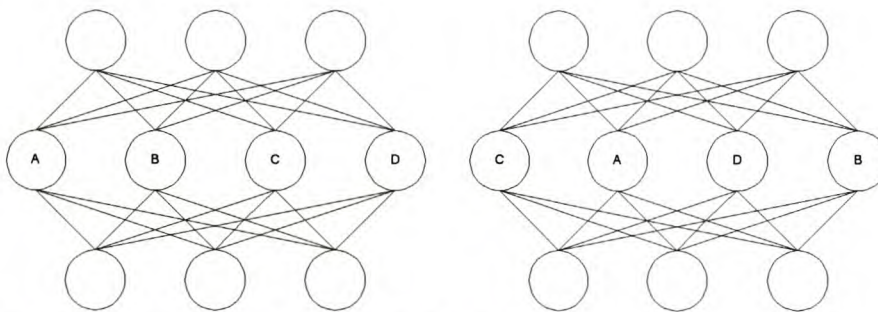


Figure 3-2 - Structural / Functional Mapping

Hyperplane samples which show considerable variance in their *fitness values* makes the search space more difficult to explore, as it provides inconsistent and conflicting feedback about whether a hyperplane presents a good region in the search space or otherwise. The structural/functional mapping problem typically slows the rate of *convergence* and makes genetic search more difficult (Whitley et al. ,1990).

3.3 MAINTAINING DIVERSITY

Evolutionary algorithms operate by continually breeding the *population's* best *individuals* to finding the best combination of the available genetic building blocks. This process typically leads to *convergence* around a single “type” of *individual*. Should a *genetic algorithm* fail to find the global optimum it is most frequently the result of premature *convergence* around such a single type of *individual*, which encodes a local optimum. Genetic *diversity* is thus a key ingredient of genetic search. Greater *diversity* in the *population* translates to more hyperplane information to drive the search. However, by increasing the representation of high performance *hyperplanes* in the *population*, *diversity* is lost. *Exploitation* has thus reduced the potential for further *exploration*. Converged *populations* are also less able to adapt to changes in the dynamic *environment*, as diverse genetic material is no longer readily available to allow for changing the solution in response to a changing problem (Whitley & Starkweather, 1990).

Local *convergence* and lack of adaptation ability have limited impact in function optimisation, but prove to be of great consequence during the evolution of complex decision strategies. An *evolutionary* algorithm in a converged *population* may only move away from suboptimal solutions, by mutating *population individuals* which results in a more random search of the solution space. This generally results in a slow and inefficient search. Maintaining a diverse *population* allows for the continued use of *recombination (crossover)* throughout, allowing for more effective traversals of the solution space. In real world problems, timely solutions to the problem is paramount. (Moriarty & Miikkulainen, 1998).

Population diversity is also essential should changes occur in the problem *domain*. In changing dynamic or unstable *environments*, strategy changes need rapid revision to avoid costly implications. A diverse *population* allows for a more adept modification of the solution strategy should changes in the *environment* dynamics occur (Moriarty & Miikkulainen, 1998).

Related to the *diversity* problem is the issue of *generalisation*. In many settings, *convergence* by the GA on a global optimum is not appropriate. In a classifier system, the GA needs to evolve a set of rules that are specialised to various tasks (or niches) rather than producing a homogeneous (converged) *population* of similar rules. A computational model of the immune system illustrates this principle; in which a *population* of antibodies needs to be evolved to recognise a set of antigens. Should the antibody *population* be sufficiently large, the evolution of different types of antibodies that specialise in recognising different classes of antigens is desirable. The evolution of one generalist antibody, that weakly matches all antigens, is necessarily an inferior solution to the problem. Should the *population* be insufficiently large to evolve a antibody *specialisation* for each type of antigen that may occur, a reasonable response would be to evolve generalists that recognise a subset or specific class of antigen. In traditional GA optimisation, the generalist usually does not arise as the *population* typically specialises in optimising a single *fitness* peak. The evolution of generalists is imperative for settings in which a single *individual* (antibody) must be able to provide more than one solution. The degree of generality required from discovered solutions should be dictated by the characteristics of the *environment* (implicitly emerge) and not be explicitly specified in the search algorithm (Smith et al., 1993).

Maintaining *diversity* in a *population* and thus controlling *convergence* remains an illusive goal. The fine tuning of the GA *operators* has not been found to be a robust solution to the controlled *convergence* problem. It has been realised that the *operators* themselves, through their mutual interaction, should provide an adaptive and optimal balance between the *exploration* and *exploitation* of the search space. The most common approaches to maintaining *diversity* involve a mechanism that will control and prevent an unbalanced proliferation of *genotypes*. Such an *operator* may be imposed through an explicit testing of *genotype diversity*, although such a mechanism may prove computationally prohibitive (Davidor, 1991).

A high rate of *mutation* or less aggressive genetic selection strategies are most commonly employed to maintain *diversity*. Increasing the *mutation* rate only succeeds in artificially introducing noise into the *population* to promote *diversity*. Less aggressive selection strategies more often only delay *convergence* at the expense of a

slower search. Despite these implications, such algorithms generally produce better results than aggressive convergent *evolutionary* algorithms (Moriarty & Miikkulainen, 1998).

A simple method of sustaining genetic *diversity* is to use large *population* sizes. Empirical tests have shown that this has the desired effect, except that it is usually necessary to double *population* size for an incremental increase in performance. This approach, however, also tends to double the search time. The importance of promoting *population diversity* has lead to the development of several more advanced methods. These include crowding, distributed *genetic algorithms*, local mating and *fitness* sharing. These techniques rely on external genetic functions to maintain *diversity* in the genetic material. *Diversity* is thus assured through the use of computationally intensive operations (Moriarty & Miikkulainen, 1998).

3.3.1 Restricting the selection process (preselection & crowding models)

Preselection and crowding attempt to maintain *population diversity* by limiting genotypically similar *individuals* from co-existing in the *population*. Preselection schemes recognise that calculating a dissimilarity index for each *individual* in the *population* would be too computationally prohibitive. As an estimate it may be assumed that a *parent* would be one of the members of the *population* closest to the new *offspring* member. Should an *offspring* member have a higher *fitness* than the worst *parent*, it replaces the *parent* (MahFoud, 1992).

Crowding is inspired by a corresponding ecological phenomenon. Similar *individuals* in a natural *population*, often of the same species, compete against each other for limited resources. Dissimilar *individuals* tend to occupy different niches which limits *competition* between dissimilar members. At equilibrium in the fixed-size *population*, new members of a particular species replace older members of that species. The overall number of members of a particular species ideally should remain constant (MahFoud, 1992).

The traditional crowding scheme is thus an elaboration on the preselection mechanism. Crowding establishes niches by replacing *individuals* that have a similar *genotype* with new *individuals*. A “steady-state” GA creates new *individuals* one at a time, which are inserted into the *population* by replacing the existing *individuals*. A subset of the *population* is selected at random (equal to the crowding factor) and the *individual* in that subset that is genotypically closest (hamming distance) to the new *individual*, is replaced. The more similar a member of the *population* becomes to other members in the *population*, the greater the selection pressure becomes against its continued survival in the *population* (MahFoud, 1992).

While it would appear that such a scheme, though computationally intensive, should be successful at maintaining *population diversity*, this is not the case in practice. Stochastic errors in the replacement of *population* members work to create significant genetic drift. Genetic drift causes a *genetic algorithm* to converge to one region of the search space, even though many different regions may be equally fit (MahFoud, 1992).

A similar approach to preselection and crowding involves a *uniqueness operator*. The uniqueness *operator* aids in maintaining *diversity* by incorporating a censorship premise. The insertion of an *offspring* into the *population* is possible only if the *offspring* is genotypically different from all members of the *population* at a specified number of loci (hamming distance) (Davidor, 1991).

3.3.2 Dividing the population into subpopulations

Several parallel GA approaches have been introduced which explicitly divide the total *population* into a number of *subpopulations* and execute the main loop of the traditional *genetic algorithm* on each *subpopulation* separately. Each *subpopulation* will, due to inherent sampling biases, exploit its genetic material in a slightly different manner and converge to a slightly different, but similarly competitive, solution. An attractive feature of this approach is that each *subpopulation* may have a different *crossover* and *mutation* rate, maintaining a balance between *exploration* and *exploitation* in a novel way. The concept stems from *population* genetics, in which

random genetic drift allows each *subpopulation* to explore a different region of the solution space, thus maintaining a diverse search preventing premature *convergence* to local optima (Tanese, 1989).

In the distributed *genetic algorithm* implementation each *subpopulation* is allowed to evolve in isolation from the others, with sporadic migration of *individuals* between *subpopulations*. Migration between *subpopulations* is intended to convey significant discoveries between *subpopulations*. The migration interval specifies the number of generations between migrations, while the migration rate specifies the number of *individuals* involved in migration. Additional *offspring* for migration are generated, followed by random selection for migration from the "overfull" *subpopulation*. As fitter *individuals* are more likely to be selected to reproduce, the proportion of the *population* containing fit *individuals* will be greater. Fitter *individuals* are thus more likely to be chosen for migration. The migrants randomly replace *individuals* in the destination *subpopulation* (Tanese, 1989). The algorithm hence becomes more adept at discovering global optima, but it has been demonstrated that even limited migration eventually leads to *convergence* and subsequent loss of *diversity* (Smith et al., 1993).

The alternative *subpopulation* implementation involves parallel *genetic algorithm* implementations without migration. Such implementations are termed partitioned *genetic algorithms*. Partitioned algorithms have been found to consistently find fitter *individuals* than the traditional *genetic algorithm* implementations. Tanese (1989), however, found that partitioned *genetic algorithms* were unable to maintain a high average *fitness* of the entire *population*, which slows the global *population* search.

3.3.3 Restricting the mating procedure (local mating)

The traditional *genetic algorithm* incorporates panmictic selection and mating, that is, any *individual* may potentially mate with any other in the *population*. Otherwise similar to crowding (which is panmictic) in approach, these methods prevent the *population* from becoming too homogeneous by largely disallowing the hybridisation effect of *crossover*.

Local mating aims to create niches in a *population* without explicitly subdividing the *population*. The reasoning for this approach lies in that an explicit subdivision of the *population*, necessarily implies a prior knowledge of the number of niches (and relative size) in the *environment*. The goal of local mating is thus also to maintain simultaneously a number of *individuals* on more than one peak in the solution landscape (Collins & Jefferson, 1991). The *population* is, for example, arranged geometrically in a two-dimensional plane with *crossover* only occurring between *individuals* that are geographical neighbours. Random genetic variation between subgroups of *individuals*, so doing allows for *exploration* of different regions of the solution space. These islands or geographical *subpopulations* gradually encode local optima as the *subpopulations* mature. As the evolution progresses certain islands of highly fit *individuals* consume lower *fitness* islands which are in their vicinity. A stepwise island growth and controlled *convergence* is maintained until the entire grid is occupied by the *individual* encoding the global optimum (Davidor, 1991). These methods thus slow *convergence* considerably, but stable *subpopulations* are not maintained (Smith et al., 1993).

3.3.4 Explicit fitness sharing

Fitness sharing establishes *subpopulations* by penalising *individuals* should other similar *individuals* exist in the *population*. This induces the establishment of uncrowded productive niches. *Fitness* sharing modifies the means by which an *individual's fitness value* is calculated, leaving the standard *genetic algorithm* itself unchanged.

In ecological theory such strategies are called negative frequency-dependent selection. *Fitness* sharing is based on the concept that *environmental* niches have finite resources available to them. As the number of *individuals* in a given niche increases, the availability of resources in the niche decreases, resulting in an effective decrease in the viability of *individuals* in the niche, and the subsequent decrease in their numbers. To maintain a viable *population* in a niche, the *population* size must come into equilibrium with the availability of resources (Smith et al., 1993).

The mechanism of *fitness* sharing cause *population* members to cluster around peaks in a search space by reducing a *population* member's *fitness* to account for the proximity of other *individuals* in the *population*. The shared *fitness* is calculated as follows -

$$f'_i = \frac{f_i}{\sum_{j=1}^N Sh(d_{ij})} \quad (3-1)$$

where d_{ij} the distance between i and j under a given metric, $Sh(d_{ij})$ is the sharing *fitness*, and σ_s and α are parameters (equation 3-2). The critical parameter in the *fitness* sharing technique is σ_s , which dictates a cut-off distance, beyond which no sharing will occur as presented in -

$$Sh(d_{ij}) = \begin{cases} 1 & \text{if } d_{ij} = 0 \\ 1 - \left(\frac{d_{ij}}{\sigma_s}\right)^\alpha & \text{if } d_{ij} \leq \sigma_s \\ 0 & \text{otherwise} \end{cases} \quad (3-2)$$

The method is robust, but has a number of limitations as presented in table 3-1. *Fitness* sharing is, however, able to establish stable niches in contrast to other methods that simply slow the approach to *convergence* (Smith et al., 1993).

Table 3-1 - Fitness sharing limitations

<i>Limitations of fitness sharing</i>
<ul style="list-style-type: none"> • A significant number of <i>fitness</i> comparisons is required which is computationally intensive (N^2 comparisons in a <i>population</i> of size N). • An appropriate setting for σ_s based on knowledge of the number of optima in the solution space is essential. • Being able to appropriately set σ_s is dependent on the presence of uniformly distributed optima in the solution space. Less uniformly distributed optima may be ignored in the search.

3.3.5 Implicit fitness sharing

Implicit fitness sharing entails the search for cooperative species which together encode the optimal solution, within a single *population* of competing and cooperating *individuals*. *Cooperation* and *competition* occur simultaneously among different *individuals* and niches or subpopulations in the total *population*. This model of intertwined, multi-level cooperative and competitive interactions to form an optimal solution appears natural, based on real-world examples of ecologies, economies, and societies. Its potential power for expressing complex concepts and behaviours, is however offset by the tremendous complexity of the *population* dynamics. Cooperative-competitive evolution isolates the most fundamental type of *cooperation*, which entails problem decomposition via *nicheing* (Horn & Goldberg, 1996).

In this model various species solve different parts of the problem, earning separate, distinct *rewards*. Each *individual* no longer represents a complete solution to the problem, but rather partial solutions that must cooperate to form a full solution. By limiting each *individuals* role in the solution, this coevolution searches for different types of *individuals* to make up the task solution. Species that compete to perform the same task compete for the same *rewards* (weak *cooperation*), while species that do not overlap in their tasks, are cooperating in an indirect manner (strong *cooperation*). The subgoal of weak *cooperation* is to exploit as much of the resources as possible. The only type of interaction between similar species is *competition* for the same resource. No comparisons are made between *individuals* as in explicit methods. The natural mechanism for handling such *competition* and encouraging search for uncovered resources is sharing of contested resources. Thus similar species share common resources by dividing them among themselves. This induces speciation or *nicheing*, an emergent phenomenon that is a prerequisite to all other types of *cooperation*. Strong *cooperation* implies successful weak *cooperation*. The implicit *nicheing* induced by the natural and simple method of sharing *reward* (credit or *fitness*) among all *individuals* who have earned it, is a powerful promoter of *cooperation*. Even under constant and rigorous (aggressive) application of *genetic algorithm operators*, such as *crossover*, *cooperation* is maintained in the *population*. The only way to maintain such high-quality *diversity* in the face of high selection pressure is to

balance *convergence* with a restorative force, such as "*niching* pressure" (Horn & Goldberg, 1996).

In a simple GA each *individual* is evaluated according to a single scalar *fitness* function independent of other *individuals* in the *population*. The simple GA's tasks may be viewed as an optimisation of the average *population fitness*. The optimum *population* thus naturally consists entirely of copies of the best *individual*. When *individuals* influence each other's *fitness evaluation*, the task of the GA may no longer be modelled as an optimisation of the total *population fitness* (Horn et al., 1994).

Utilising implicit *niching* it may be demonstrated that a dynamic equilibrium point in the *population* is reached quickly (a fast *convergence* rate) and that the equilibrium is maintained (high restorative pressure causes long niche extinction times). The ability to balance selection pressure with restorative force allows for the maintenance of high quality, diverse niches virtually indefinitely (Horn & Goldberg, 1996).

As the search entails the optimisation of different aspects of the problem, several parallel searches are performed in decompositions of the solution space. This allows for the more aggressive application of the *genetic algorithm*. This allows for a marked increase in the rate of evolution. The *evolutionary* algorithm utilised in this study, Symbiotic Adaptive Neuro Evolution (*SANE*), largely falls into this category of *evolutionary* algorithms, with modification that allows for the evolution of neural *networks* (Moriarty & Miikkulainen, 1998).

3.4 COMPARING THE SANE APPROACH TO OTHER NEURAL NETWORK APPROACHES

The use of artificial evolution has become a popular method for developing control policies in complex problem *domains*. This development requires an approach with marked differences from function optimisation *evolutionary* algorithm approaches. Particularly, in real world applications the available number of *fitness evaluations* are limited, as compared to the often limitless number of *fitness evaluations* allowed in function optimisation problems (Moriarty & Miikkulainen, 1998).

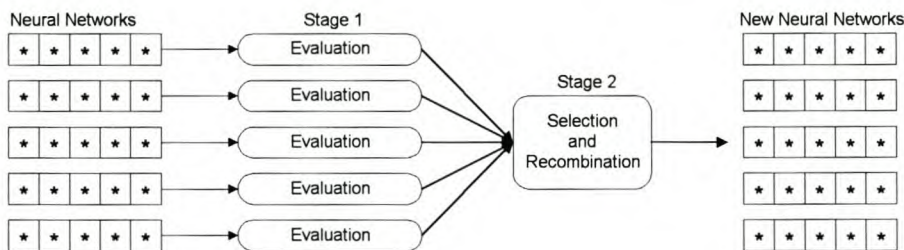
Cooperative *Coevolutionary* algorithms present a promising alternative for evolving complex dynamic control policies. In cooperative *coevolutionary* algorithms each *individual* only represents a partial solution to the problem at hand. Complete problem solutions are compiled by grouping selections of these partial solutions together. This approach thus attempts to optimally combine solution building blocks (partial solutions) with other partial solutions to form a full solution. Several parallel searches for different aspects of the solution should prove to be more effective than a single search for the entire solution. Maintaining these partial solutions in a single *population* avoids *convergence* of the *population* to a single *individual*. *Diversity* is thus maintained throughout (Moriarty & Miikkulainen, 1998).

Symbiotic, Adaptive Neuro-Evolution (*SANE*) was designed as an efficient method for developing *neurocontrollers* in dynamic *environments*. Whereas most neuro-evolutionary techniques (i.e., GENITOR) operate on a *population* of neural *networks*, *SANE* evolves a *population* of *neurons*. Each *individual neuron*'s task is to determine the appropriate connections and *weights* that together with other *neurons* form a robust *neurocontroller*. *SANE* allows for more accurate searching of the dynamic *domain*, as it recognises that *neurons* are the basic building blocks of neural *networks*. As no single *neuron* may represent a single solution, *evolutionary* pressure is introduced to evolve *neuron specialisations* maintaining *diversity* (Moriarty & Miikkulainen, 1998).

In the more common approach to neuro-evolution, each *individual* represents a complete *network* that is independently evaluated from the other *networks* in the *population*. In the GENITOR II algorithm developed by Whitley et al. (1993), which has been implemented in a wide variety of applications, each *individual* encodes a full neural *network* (complete solution). GENITOR II incorporates adaptive *mutation* (as *population diversity* is lost, the *mutation* rate increases proportionally) and a distributed genetic search (section 3.3.2) to sustain genetic *diversity*. As the *population* becomes increasingly homogeneous due to *convergence*, the rate of *mutation* is increased to maintain diverse genetic material in the *population*. The *population diversity* is indirectly monitored throughout evolution by measuring the hamming distance between two *parents* during reproduction, which may be computationally intensive. The more alike the two *individuals* are, the greater the

probability of *mutation* being applied in the *offspring*. Also, a distributed scheme is incorporated to aggressively exploit *subpopulations* without exhausting the *diversity* in the entire *population* (Whitley et al., 1993). The full and partial neural *network encoding* schemes were empirically evaluated (Moriarty & Miikkulainen, 1996), with *SANE (implicit fitness sharing)* being superior to GENITOR II (adaptive *mutation*, distributed algorithm) in performance. The different strategies for *neurocontroller* evolution are illustrated in figure 3-3.

(a)



(b)

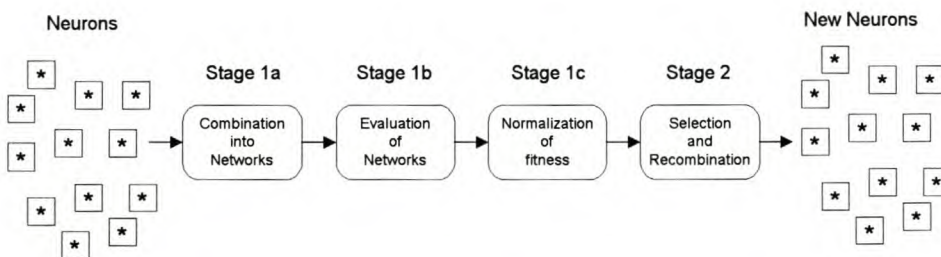


Figure 3-3 - An illustration of the neuro-evolution performed in SANE (b) compared to the standard approach to neuro-evolution (a).

By treating each *individual* as a separate, full solution the algorithm focuses the search towards a single dominant *individual*. In the approach followed by *SANE* evolutionary pressure exists to form symbiotic relationships. Symbiotic evolution may be defined as a type of evolution where *individuals* explicitly cooperate in order to survive in the *population*. In this respect symbiotic evolution is distinct from other *coevolutionary* (coadaptive) algorithms, in which *individuals* compete rather than cooperate with each other for survival. In the computational model of the immune system (Smith et al., 1993), each antibody must compete for survival with other antibodies in the *subpopulation* to recognise a given antigen. In the immune system model the *fitness* of each *individual* reflects how effectively it matches its opposing

antigen, not how effectively it cooperates with other *individuals*. The antibodies are not dependent on other antibodies for recognition of an antigen and only interact implicitly through *competition*. The maintenance of niches in the *population* is thus based on weak *cooperation* (coadaptation) and not on strong *cooperation* (symbiotic evolution) (Moriarty & Miikkulainen, 1998).

Neuron specialisation ensures *diversity*. A single *neuron* cannot dominate a *population*, since highest *fitness* may only be obtained in a *population* where other *specialisations* are present. Should a *specialisation* become too prevalent, its members will not be effectively combined with other *specialisations*. Redundant partial solutions are consequently ineffectively combined with other *specialisations* and thus obtain lower *fitness evaluations*. This *evolutionary* pressure thus selects against *individuals* of dominant *specialisations*. Solutions are thus found in diverse unconverged *populations* (Moriarty & Miikkulainen, 1998).

Evolution at the *neuron* level more accurately evaluates the genetic building blocks. In the more prevalent (common) *network*-level evolution, each *neuron* is implemented with the other *neurons* encoded on its particular chromosome. This type of genetic representation may lead to an effective *neuron* existing along with ineffective *neurons*. Due to the resulting low *fitness evaluation* the knowledge contained in the effective *neuron* may be lost. In *neuron* level evolution, the continual *recombination* of a *neuron* with many other *neurons* results in a more accurate *evaluation* of the neural genetic building blocks (Moriarty & Miikkulainen, 1998).

SANE also takes advantage of *a priori* knowledge that neural *networks* are composed of *individual neurons*. *Neuron*-level evolution explicitly promotes genetic material in the *population* that may be useful in constructing complete neural *networks*. A *network*-level approach is implicit. Evolving at the *neuron* level the *evolutionary* algorithm is no longer expected to identify *neurons* as the significant building blocks, as *neurons* are the focus of the evolution (Moriarty & Miikkulainen, 1998).

3.5 MAINTAINING EFFECTIVE NEURON COLLECTIONS

Neuron evolution alone has, however, proved to be insufficiently powerful in generating neural *networks* for complex tasks. Moriarty & Mikkulainen (1996b) enhanced the general principle of *implicit fitness sharing* (Horn et al., 1994), by investigating how partial solutions may be effectively combined to form complete solutions (Moriarty & Mikkulainen, 1998).

Knowledge of the useful combinations of *neurons* must be preserved and exploited. Intelligent direction for *neuron* (partial solution) combination is desirable. Without intelligent *recombination*, *neurons* may not be combined with other *neurons* that effectively cooperate. An effective *neuron* may otherwise be lost as it may not be effectively combined during a generation. The quality of randomly combined *networks* also vary to a large extent during evolution. In the initial generations this is advantageous, as many different kinds of *networks* are evaluated to find cooperative *neurons*. Without intelligent *neuron* combination the direction of the search during later generations is often stalled, due to continued inconsistent combination of *neurons* when the desired response is to focus on better performing *networks* (Moriarty & Mikkulainen, 1998).

To overcome this difficulty memory of past effective *networks* (effective *neuron* combinations) needs to be maintained. A mechanism is introduced in *SANE* which involves evolving a layer of neural *network* blueprints along with the *neuron* evolution. This blueprint *population* carries knowledge of effective *neuron* combinations, that may be effectively used in the subsequent generation. The blueprint *population* results in more accurate *neuron evaluations*, as *neurons* are more consistently combined with other *neurons* that have cooperated well together. Fitter *neurons* also garner more pointers from the blueprint *population*, thereby participating in a larger number of *networks*. This biasing towards historically more cooperative *neurons* provides more accurate *evaluation* of the top *neurons*. Newer *neurons* in this algorithm model, however, may consequently receive too few *evaluations* for proper *evaluation*. Empirically it has been observed that this allocation of more trials to the top *neurons*, performs better than uniform *neuron* participation. Figure 3-4 illustrates

the relationship between the blueprint and the *neuron population* (Moriarty & Mikkulainen, 1998).

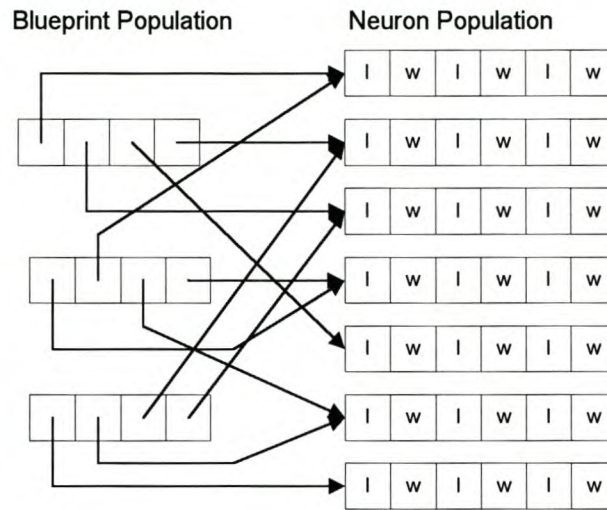


Figure 3-4 - Network blueprint population in relation to the neuron population.

The primary advantage of evolving *network* blueprints is the *exploitation* of the best *networks* found during evolution. By evolving the blueprint *population*, the best *neuron* combinations are also recombined to form new, potentially better, collections of *neurons*. The blueprint level evolution thus provides a very exploitative search that utilises the best *neuron* combinations found, thereby focusing the search in later generations (Moriarty & Mikkulainen, 1998).

3.6 SANE IMPLEMENTATION

For the purposes of this study, the *SANE* algorithm was implemented in a Windows *environment*, Delphi 4.0, with the various units described in appendix B and the source code contained on the included CD. *SANE* evolves both a *neuron* and a *network blueprint population*. Each *neuron* in the hidden layer specifies a set of *weights* and connections to the input layer and output layer. Each *individual* in the *network population* specifies a grouping of *neurons* to include in the *network* to be evaluated. The *neuron* evolution thus searches for effective partial solutions (*neurons*), whilst the blueprint evolution searches for effective combinations of these partial solutions (*networks*) (Moriarty & Mikkulainen, 1998).

Each *neuron* represents a hidden *neuron* in a 3-layer partially connected feed-forward *network*. The *neuron individuals'* genes encode a series of connections and *weights*. Each *neuron* is a series represented by an even number of *gene* pairs. The first (even series position) *gene* in a pair encodes a *neuron* connections and the second (uneven series position) *gene* in a pair, encodes the *weight* for that particular connection. Each connection *gene* is an integer value that range between 0 and one less than the total number of input and output nodes. The decoding of the connection *gene* results in the assignment of a connection to either an input or an output node. Should the integer value in the connection field be less than the total number of input nodes, the connection is made to the corresponding input node number. Otherwise, the connection is made to the corresponding output node number. Connections are thus probabilistically assigned to either input or output nodes, based on the number ratio of input to output nodes. Each *weight gene* is a floating point value with a *gaussian* distribution around the mean 0, with a standard deviation of typically 2. Initially the connection and *weight* genes are randomly allocated for each *neuron* in the *population* (Moriarty & Miikkulainen, 1998).

Each *individual* in the blueprint *population* is comprised of a set of *neuron* pointers (address pointers) to *neuron* structures. The number of *neurons* in each *network* is fixed, depending on the complexity of the problem to be solved. Initially the *neuron* address pointers are assigned at random to *neuron* structures (Moriarty & Miikkulainen, 1998).

SANE's evolutionary generational algorithm operates in two main phases – *evaluation* and *recombination*. The macro flow chart of the *SANE* algorithm is illustrated in figure 3-5 (Moriarty & Miikkulainen, 1998).

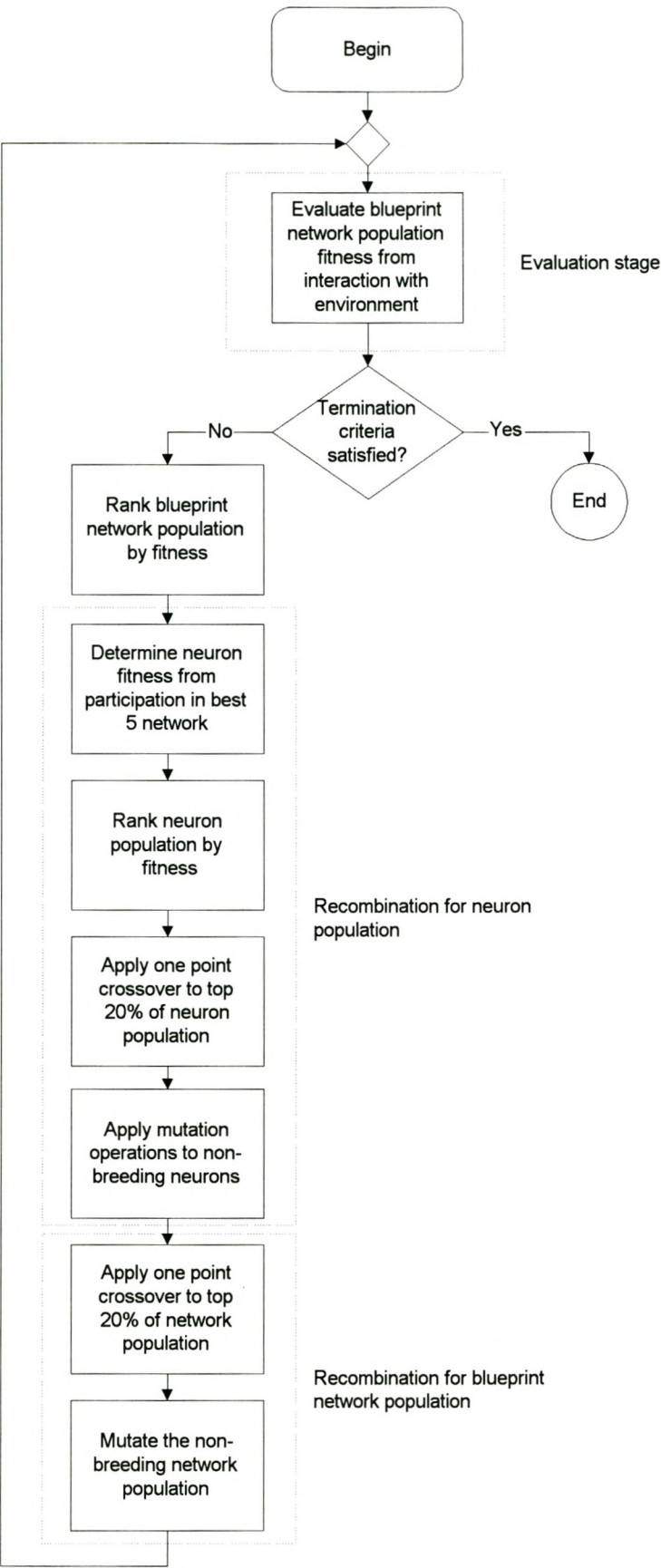


Figure 3-5 - Macro flow chart of the SANE algorithm

3.6.1 Evaluation stage

The *evaluation* phase facilitates the determination of the *fitness* of each *neuron* and *network* in the *populations*. Each blueprint is used to select *neuron* subpopulations of size, ζ , to form a neural *network*.

Network blueprints are evaluated based on reinforcement from direct interaction with either a dynamic simulation or real world *environment* (section 2.3). Figure 3-6 illustrates the process of *evaluation* for a single *individual*. The plant *state* is initially typically set to an initial *state* that lies in the region around the desired set points. The dynamic *state* equations are solved, using a fourth order Runge-Kutta for simultaneous solution, for the duration of a single sample period. The plant enters a new *state* and the *state* variables and other inputs to the *neurocontroller* determine the *neurocontroller's* control action. This control action will determine the solution of the *state* equations for the next sample period. The error from the desired *state* for the current sample period is calculated and stored for *fitness* calculation purposes. The current plant *state* is evaluated to determine if a *state* has been entered that requires premature failure of the *evaluation* trial. Should premature failure occur, a specified maximum error is assigned to the remaining time steps (sample periods) for the trial. Should premature failure not occur, the described sequence of events is repeated, until the *evaluation* time has ended. A *fitness* value is assigned to the blueprint *network* based on the criteria specified by the *fitness* function.

Each *individual neuron* pointed to by a blueprint *network* is assigned a *fitness* based on the average *fitness* of the best five *networks* that it participated in. Utilising only the best five *networks* provides a driving force which discourages against the selection of *neurons* that are critical in the best *networks*, but ineffective in non-performing *networks* (Moriarty & Miikkulainen, 1998).

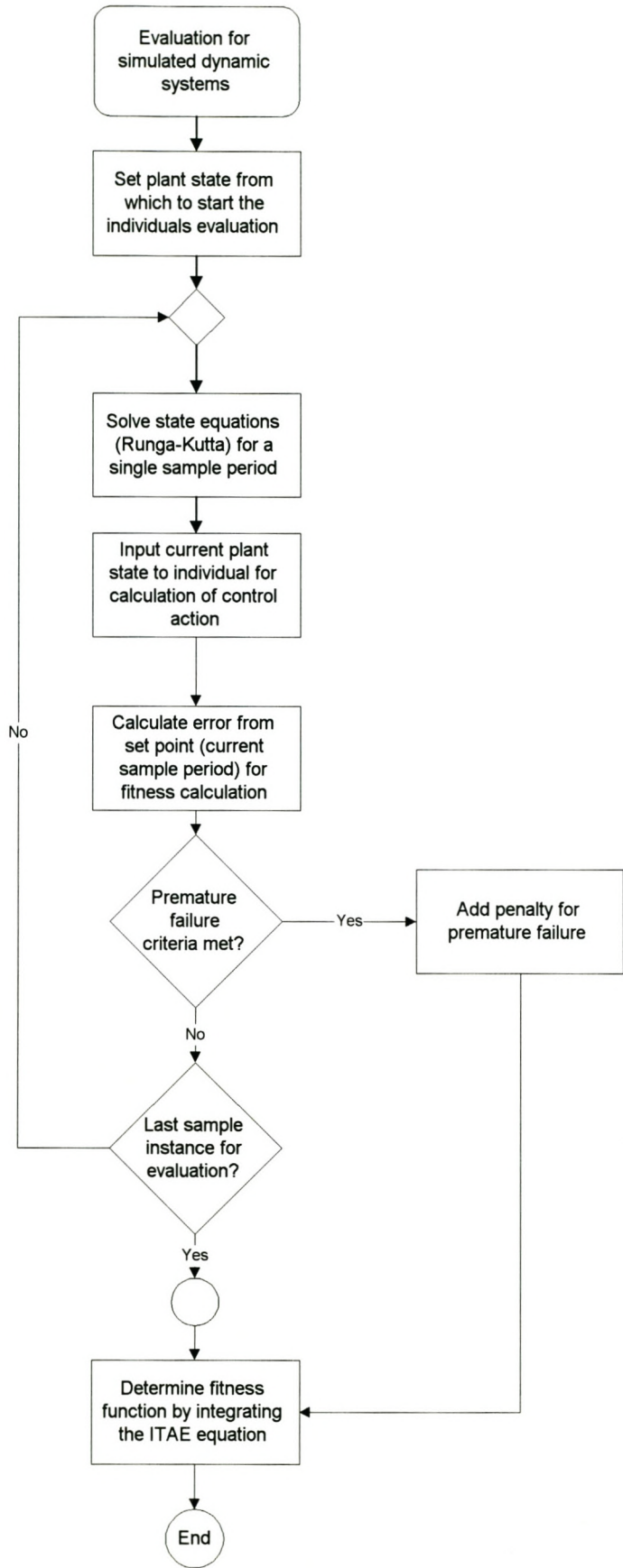


Figure 3-6 - Micro flow chart for evaluation phase for a single individual

3.6.2 Recombination for the neuron population

After *evaluation* the *neuron* and the *network blueprint population* are ranked based on the assigned *fitness*. For each *neuron* in typically the top 20% of the *neuron population* (*elite population*), a mate is selected randomly from the *neurons* that have a higher *fitness* value than the selected *neuron's fitness*. Thus, the *neuron* ranked 3rd may only reproduce with the *neurons* ranked 2nd and first. Two *offspring neurons* are created from a one-point *crossover operator*. One of the *offspring neurons* produced by *crossover* is randomly selected to enter the *population*. The other *offspring neuron* is replaced randomly by one of the *parent neurons*, after which the *offspring neuron* is inserted into the *population*. Copying one of the *parent neurons* as the second *offspring* reduces the effect of adverse *neuron mutation* in the *blueprint population*. The two *offspring* replace the most ineffective *neurons* in the *population* according to rank. This results in a number of the most ineffective *neurons* (double that of the *elite neurons*) being replaced in the *population* for each generation. No *mutation operator* is used on the *elite* or breeding *neurons* (Moriarty & Miikkulainen, 1998).

Only the non-*elite* (non-breeding) portion of the *population* partakes in *neuron mutation*. For connection (even position) genes a 2% probability exists that a connection may be randomly reassigned to either an input or output node. For *weight* (uneven) genes a *mutation* probability of 4% exist for a random *gaussian weight* adjustment and a 0.1% probability for a *weight* sign inversion. A random *gaussian* adjustment results in a modification of the original *weight*, within a random *gaussian* distribution with a standard deviation of 1 (original *weight* is mean) (Moriarty & Miikkulainen, 1998).

This aggressive, elitist breeding strategy is normally not incorporated in *neurocontrollers* evolution, as this would generally lead to premature *convergence* of the *population*. As *SANE* provides for pressure against *convergence*, *SANE* performs well with this aggressive strategy (Moriarty & Miikkulainen, 1998).

3.6.3 Recombination for the network population

The *crossover* operation in the blueprint *population* results in the exchange of address pointers to *neurons* in the *neuron population*. Should a *parent* point to a specific *neuron*, one of its *children* will consequently also point to that particular *neuron* (Moriarty & Miikkulainen, 1998).

To avoid *convergence* in the blueprint *population*, a twofold *mutation* strategy is incorporated. A pointer in each *offspring* blueprint is randomly reassigned at a rate of 0.2% to a *neuron* in the *neuron population*. This promotes the use of other *neurons*, other than the *neurons* in the top *neuron* subpopulation. A *neuron* that does not participate in any *networks* may so doing obtain a pointer from the blueprint *population*. The *neuron* pointers in the top blueprints are always preserved, as *mutation* does not occur in breeding (*elite*) *networks* (Moriarty & Miikkulainen, 1998).

Each *offspring neuron* produced by *crossover* in the *neuron population* is potentially better than or an exact copy of its *parent neuron*. The blueprint evolution takes advantage of this knowledge, by reassigning breeding *neuron* pointers to *offspring neuron* pointers. This is accomplished with a 50% probability. This *mutation* function is only performed on the *offspring* blueprints, thus preserving the *parent* blueprints (Moriarty & Miikkulainen, 1998).

As pointers are occasionally reassigned to *offspring neurons*, new *neuron* structures may be evaluated. As *neuron* pointers are so doing also reassigned to exact copies of *parent neurons*, some resilience is incorporated into the algorithm against adverse *mutation* at the *neuron* level. Should pointers not be reassigned to copies, several blueprint *networks* may point to the same *neuron*. Any *mutation* in that *neuron* would consequently effect each *network* that points to it. This strategy limits this possible adverse effect to only a few blueprint *networks*. This is similar to schema promotion in standard *evolutionary* algorithms. As evolution progresses, highly fit *schemata* (*neurons*) become more prevalent in the *population*. *Mutation* to one copy of the *schemata* does not affect other copies in the *population* (Moriarty & Miikkulainen, 1998).

3.7 THE SUITABILITY OF SANE FOR EVOLVING NEURAL NETWORKS

In section 3.2 several complications are discussed that result when neural *networks* are evolved using *genetic algorithms*. These include scale-up and the structural/functional mapping problem.

The evolution of large complete neural *networks* (i.e., GENITOR II) requires binary *encodings* that have long *genotype* (string) *encodings*. The longer the encoded solution the greater the probability of the *crossover operator* becoming disruptive to the genetic search (section 3.2). *SANE* deals with this scale-up difficulty by *encoding* the *neuron population* utilising real value *encoding* (Whitley & Starkweather, 1990) as described in section 3.5. This reduces the *genotype* encoded length and consequently results in more effective usage of the *crossover operator*. The evolution of *individual neurons*, instead of complete *networks*, also reduces the probability of disruptive *crossover* occurring. *Individual neurons* have far shorter *encodings* than full *network encodings*, thus reducing the probability of relevant schema being separated by great distances in the *neuron encoding*.

The *neuron population* also effectively deals with the structural/functional mapping problem. As each *neuron* typically represents a single functionality to the full task, the structural/functional mapping problem cannot result, as occurs when the *crossover operator* is applied to fully encoded neural *networks* (section 3.2). Loss of functionality due to the *crossover operator* is thus not a concern at the *neuron* level. *Crossover* between two functionally different *neurons* that cooperate well, may result in two tasks being effectively combined into a single *neuron*. *Neuron crossover* thus promotes the emergence of generalists (section 3.3) as described by Smith et al. (1993), in an *environment* where resource and *network* size is limited. The top *neurons* in each generation are also copied unchanged to the next generation. Any detrimental *crossover* or *mutation* effects thus only effects the *offspring neurons*. A robust search is thus maintained at the *neuron* level.

The *network* (blueprint) *population* as implemented in *SANE* is, however, susceptible to the structural/functional mapping problem (section 3.2). Blueprint *networks* point

to functional *neurons* and *crossover* may disrupt useful combinations, should two important cooperating functionally different *neurons* reside at each end of the blueprint *network encoding*. *SANE* deals with this deficiency indirectly, by copying the top *networks* unchanged to the next generation. Should poor *offspring* have resulted in the previous generation, the effect may only be to slow the genetic search as a result of producing poor *offspring* from two effective *parents*. The resulting slowed *convergence* to an optimal solution is due to the disruption in the premise of the *genetic algorithm* search strategy; which probabilistically relies on *offspring* being potentially better than their *parents*. A slower *convergence* rate to an optimal solution remains detrimental in real world applications, as a slower search implies utilising more *evaluations* which may prove costly in practice.

3.8 PAST APPLICATIONS OF SANE

3.8.1 Game Tree Search

The standard method for game tree searching is a minimax search. Minimax searches, however, rely on heuristic *evaluation* functions that often prove to be inaccurate. The searches generate errors that are propagated through the tree and may result in the selection of suboptimal moves. *SANE* has been implemented to function as a filter for minimax searches, effectively only making information available that will result in good decisions. *SANE* was implemented in a world champion Othello program, "Bill". The program's performance was significantly improved (Moriarty & Mikkulainen, 1998).

3.8.2 Controlling Chaos

Chaotic behaviour in dynamic systems may be suppressed by periodically applying small, carefully chosen perturbations. This is done with the goal to stabilise an unstable periodic orbit of the system. *SANE* has successfully been applied in achieving robust control. In contrast to other methods for the control of chaotic systems, no assumptions need be made concerning the system – in particular, no knowledge is required of the dimensionality of the dynamics or the location of any

unstable fixed points. As in other implementations of *SANE*, the controller structure is not fixed and the *neurocontroller* is free to adopt nonlinear forms. Other methods often require careful study to choose the structure of the *network* for a given system. In the study by Weeks & Burgess (1997) the particular desired characteristics of the neural *network* relates to the algorithm's ability to stabilise unstable fixed points of a Poincaré map. The goal of the algorithm is thus to find a *neurocontroller* that will drive a chaotic system to eventually reach a fixed point.

Weeks and Burgess (1997) found that certain characteristics of the developed *neurocontrollers* were not reproduced by other control methods. It was found that the *neurocontrollers* were able to drive the system to the fixed point, even when starting far from the fixed point. Other controllers typically only succeed when activated whilst the system is near the fixed point. The *neurocontrollers* were thus able to generalise over a larger solution *state* space.

Weeks and Burgess (1997) made a number of observations relevant to this study -

- The proper choice of *fitness* function determines the speed with which the EA converges on the optimum solution.
- The *fitness* function could be biased to allow for *networks* that produced desired behaviour. Although it is possible to control chaotic systems by allowing large perturbations, small perturbations are generally required to accomplish the task near the fixed point. A greater *reward* could be awarded to *neurocontrollers* that accomplished the task using small perturbations.
- As their chaotic systems contained significant noise, the same *neurocontroller* could be tested numerous times with different results. To allow for robust *fitness evaluation* in the presence of noise, each *network* was evaluated twice and assigned the average *fitness*.
- Allowing more input nodes than required to control the system slowed the evolution slightly, but *SANE* learned to ignore superfluous inputs to the *network*. *SANE* appears to be fairly robust as far as an over specified input layer is concerned.

- The results also appeared to be robust to changes in the number of *neurons* in the hidden layer. Empirically, it was found that the evolution proceeded fastest when the number of hidden layer *neurons* was approximately twice the number of input *neurons*. This demonstrated the strength of *SANE* to make redundant superfluous *neurons* in the hidden layer. As the required number of hidden *neurons* is often unknown, this proves to be a useful feature.
- Trials with different output layer arrangements were, however, unsuccessful and often could not control the system at all.
- Weeks and Burgess (1997) found their results to be fairly independent of *population* size in their directed *SANE* algorithm. They concluded that both *mutation* and *crossover* were important for successful evolution, as a good solution could typically not be found when omitting one of these genetic *operators*.
- Some concern was expressed regarding the number of *evaluations* required to develop an effective *neurocontroller*, should on-line training be desirable. It was concluded that including more information about the physical system (such as the location of the unstable fixed point) to be controlled within the algorithm, would allow the training process to find an effective *neurocontroller* in far fewer generations. Also, pretraining of the *network* and *neuron populations* on modelled physical processes (even imperfect models) will significantly speed the evolution.

3.8.3 Robot Arm Control

Most neural *network* applications to robot arm control learn hand-eye coordination through supervised training methods such as back-propagation or conjugate gradient descent. Supervised learning, however, requires training examples that demonstrate correct mappings from input to output. The current approach for generating training examples for robot arm control are limited and ineffective in uncertain or obstacle-filled *environments*. *Neurocontrollers* that learn from supervised techniques cannot integrate target reaching with obstacle avoidance. *SANE*, however, does not require input/output examples and can learn the intermediate joint rotations necessary for avoiding obstacles through trial and error experimentation (Moriarty & Miikkulainen, 1998).

3.9 REINFORCEMENT LEARNING AS A STRATEGY TO PROCESS CONTROL DESIGN

Reinforcement learning techniques have most frequently been applied in areas such as robotics and game theory. Despite obvious similarities, process control and robotic control differ in several important ways. Chemical plants are generally operated continuously, and try to respond to and minimise the effect of disturbances, rather than (for example) follow specified trajectories. Further research is needed in devising more powerful methods of learning models of dynamic systems (or learning to correct models derived from first principles so that they describe real plants) and in developing *reinforcement learning* schemes appropriate to control applications. Algorithms must be modified to handle essentially infinite sequences of data, and to overcome problems such as "forgetting", which may occur should a particular operating condition be encountered infrequently during learning - the memory of applicable actions is lost due to infrequent reinforcement of that *state*. Real-world problems with variable time lags and large amounts of data add to the challenge. Learning speed presents a significant challenge in real-time applications (Ungar, 1991).

Some aspects of chemical process control are simpler than robotic control. Chemical process control problems do not have the complicating questions of visual interpretation common to robotics. Fairly accurate (although sometimes erroneous) sensor readings of temperatures and pressures are virtually continuously available and a fairly clear identification and control problem may subsequently be posed. The abundance and frequency, in itself, of measurements poses a challenge. The effective use of a vast amount of available process data to improve the model or present control of the process, is of particular relevance. The goal of chemical process control is also quantitative (maximum production of a chemical) rather than qualitative (avoiding obstacles) (Ungar, 1991).

Much of the difficulty in controlling any process lies in the complexity of the process being controlled. This complexity may arise in several ways. Highly nonlinear systems are difficult to control, particularly when they have complex dynamics (such

as instabilities to limit cycles and chaos). Difficulties may often be presented by constraints, either in control parameters (e.g., there is a maximum rate at which the system may be heated) or in the operating regime (e.g., heating above a certain point leads to a runaway reaction). Lack of exact knowledge of the process naturally also makes control more difficult (Ungar, 1991).

Some of the above challenges are also common to robotics control problems. Difficulties such as significant lag times between the time a control action is taken and the time a response is observed are more characteristic of chemical process control. The presence of response delays result in the model of the dynamic system not being invertible. Time delay also presents a temporal *credit assignment* problem (section 2.5). It may not be trivial to determine how *reward* should be credited to a given control action. Many chemical processes also have a spatial *credit assignment* problem - systems may have many sensors and controllers (are multiple-input-multiple-output, or *MIMO*), and it may not be clear in which configuration to connect sensors and controllers or how changes to multiple controllers interact (Ungar, 1991).

Optimal control of many chemical plants also requires systems which make use of predictions of future plant behaviour. This can occur in time-varying processes such as batch processes, where optimisation over time is desired, or it may occur in fairly simple continuous systems, where nonlinearities may cause an inverse response (a change in the control parameter that may initially move the process in a direction opposite to its longer term effect) (Ungar, 1991).

Chemical processes are also different from robotics control problems in that experimentation must necessarily be limited and conservative. Although it may be feasible to have a robot attempt, but fail, fifty times at performing some task; it is not acceptable to have a reactor attempt, but fail, fifty times at producing product. Unlike many "toy" problems, numerous *evaluations* cannot be afforded for long periods of time prior to achieve good performance. This means - among other consequences - starting from models based on first principles and improving with a relatively conservative learning algorithm (Ungar, 1991).

As the *SANE* algorithm has mainly been utilised for applications that share similar challenges to those in robotic control (section 3.8), the feasibility of *SANE* in process control applications needed to be determined. The classic inverted pendulum problem is considered (Appendix A), in order to evaluate *SANE*'s ability to discover control policies that adhere to a specified desired closed loop response. The inverted pendulum problem has historically been a benchmark problem for evaluating *reinforcement learning* algorithms, but typically only from a controllability consideration. A closed loop performance based implementation is described in appendix A, in which *SANE* needs to discover a desired closed loop response for the system. A comparison between controllers developed with modern *state* space design and the *SANE* algorithm is also presented. The *SANE* solution was found to generalise effectively in the presence of model uncertainty, making a single controller applicable over a broad range of model parameters. Robustness in the face of uncertainty, which is prevalent in process control implementations, is an extremely desirable feature. Consideration is also given (Appendix A) to analysing *SANE*'s ability to deal with *reinforcement learning* challenges, such as non-*Markov* decision processes and the presence of sensor noise during learning. *SANE* was found to perform robustly while learning with the presence of sensor noise, but had limited success with non-*Markov* implementations of the inverted pendulum problem (Appendix A). Based on this analysis of *SANE*'s potential in learning to control a continuous, unstable multi-input multi-output (*MIMO*) system to a desired closed loop response, the *SANE* algorithm was deemed applicable to solving the particular challenges posed by process control environments.

3.10 CONCLUDING REMARKS

Chapter 3 describes challenges faced when evolving neural *network* structures. The importance of maintaining *population diversity* in genetic search when searching for complex behavioural solutions, is also stressed. *SANE* was introduced as a novel approach to *neurocontroller* design, by implicitly allowing for the emergence of cooperative species (or *nicheing*) to form complete solutions to complex problems. The significant success that has been achieved with *SANE* in a variety of applications, such as robotics, game theory and in controlling chaos; provides empirical motivation

that the *SANE* algorithm may be utilized for an even wider variety of problem definitions, such as process control applications. Application to process control problems, however, introduces many new challenges not encountered in more traditional *reinforcement learning* applications (i.e., robotics), such as process response time delays. This complicates the *credit assignment* problem.

3.11 SYMBOLS FOR CHAPTER 3

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
d	difference measure between two <i>individuals</i> in <i>population</i>	-
f	<i>individual's fitness</i> in <i>fitness</i> sharing	-
L	Length of a binary bit string	-
N	Number of <i>fitness evaluations</i> for <i>fitness</i> sharing	-
Sh	Sharing <i>fitness</i>	-

Greek symbols

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
α	<i>Fitness</i> sharing parameter	-
σ_s	<i>fitness</i> sharing parameter	-
ζ	Number of <i>neurons</i> forming a blueprint <i>network</i>	-

Subscript

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
i, j	<i>Individual</i> in <i>genetic algorithm</i> <i>population</i>	-

4 NEURAL NETWORKS FOR CONTROL SYSTEMS

OBJECTIVES OF CHAPTER 4

- Demonstrate the control system configurations in which neural *networks* have found application.
- Illustrate the advantages and disadvantage of each approach in addressing control problems within a neural *network* framework.

4.1 NEURAL NETWORK CONTROL STRUCTURES

Neural *networks* have found utilisation in many areas related to process control (table 4-1), as neural *networks* possess characteristics advantageous to control (table 4-2). In process control applications neural *networks* may be incorporated into the control loop in either direct or indirect control methods. In the direct method, a neural *network* is trained with observed input-output data from the system to represent the system's inverse dynamics. The resulting controller is used in a feedforward fashion. In the indirect method, the neural *network* is trained with input-output data from the dynamic system to represent the forward dynamics. Given the current *state* and the current control action, the *network* learns to predict the next *state* of the system. This process model may consequently be used by a control algorithm to calculate the control action (Psichogios & Ungar, 1991).

Table 4-1 - Neural networks in control (Stephanopoulos & Han, 1996)

Neural networks in control

- Representation of a nonlinear input-output model.
- Function as the control law
- Supervisory classifier for controller performance *evaluation* and diagnoses of failures or the need for tuning.
- Present a switching logic that guides the adaptation of the controller's structure

Despite possessing characteristics favourable to process control, the implementation of neural *networks* in control loops poses difficulties. When employing a control scheme using neural *networks* as the process model, there are numerous challenges in obtaining effective data to train the neural *network*. Generally, data is obtained by making various random changes in process inputs over the whole operation range. Perturbation of process inputs is, however, practically prohibited especially in chemical processes, due to economic loss and safety considerations. Additionally, as chemical processes consist of many sequential sub-units, the effects of such perturbations may be propagated to downstream units. Should large scale plants need to be disturbed significantly for neural *network* training, the economic loss may be significant (Kim et al., 1997).

Training data may, however, also be obtained from historical operation data. As most industrial plants are operated by simple linear controllers, sufficient sequences of changes to control actions for obtaining effective training data, are, however, not typically contained in historical plant data. Satisfactory neural models may not be obtainable from past process data (Kim et al., 1997).

Table 4-2 - Neural *network* characteristics relevant to control systems (Kim et al., 1997)

<i>Relevant control features of neural networks</i>
<ul style="list-style-type: none"> • Ability to represent arbitrary non-linear relations. • Capability of learning uncertain systems through both off-line and on-line <i>weight</i> adaptation. • Flexibility to handle the input information which is transformed to internal representation allowing data fusion, with both quantitative and qualitative signals. • Parallel distributed processing architecture allowing fast processing for large scale dynamic systems. • Neural <i>network</i> architecture provides a degree of robustness through fault tolerance and graceful degradation.

The control literature presents a number of structures or methods for the control of nonlinear systems. Hunt et al. (1992) focused on those structures that have a direct reliance on system forward and inverse models. These structures are, from a

mainstream control theory viewpoint, well-established and their properties have been extensively analysed.

Hunt et al. (1992) classified the control structures using neural *networks* as supervised control, direct inverse control, model reference control, *Internal Model Control*, predictive control and gain scheduling.

4.1.1 Supervised Control

In some control tasks a human provides the feedback control action, especially where it has been difficult to design an automatic controller using standard control techniques. It may be impossible to obtain an analytical model of the controlled system. Should an automatic controller mimic the actions of a human *operator*, it is designated as a supervised controller. A neural *network* or an expert system may provide the knowledge representation and control formalism required. Training such a neural *network* involves providing the *network* with the same sensory input information that a human may receive. The *network* target outputs used for training correspond to a human control input to the system (Hunt et al., 1992).

4.1.2 Direct Inverse Control

Direct inverse control (figure 4-1) utilises an inverse system model. The inverse model is cascaded with the process, so that the composed system results in an identity mapping between desired response (i.e. the *network* inputs) and the controlled system output. The *network* thus acts directly as the controller in such a configuration.

This approach relies significantly on the fidelity of the inverse model used as the controller. The robustness of such controllers is a paramount concern in their application. A lack of robustness has primarily been attributed to a lack of feedback to the controller. On-line learning may alleviate this robustness concern, by adjusting the inverse of the model on-line (Hunt et al., 1992).

Kim et al. (1997) also allude to learning stability difficulties and a lack of robustness that arises in the inverse control scheme. Chemical plants may be operated at high pressure and/or temperature, thus the control scheme must guarantee closed-loop stability. To improve robustness, a feedback controller may be included in the direct inverse control scheme.

Furthermore, Stephanopoulos & Han (1996) indicate that there is a growing body of evidence that the use of neural networks to map directly the process inverse from operating data is brittle and prone to failure.

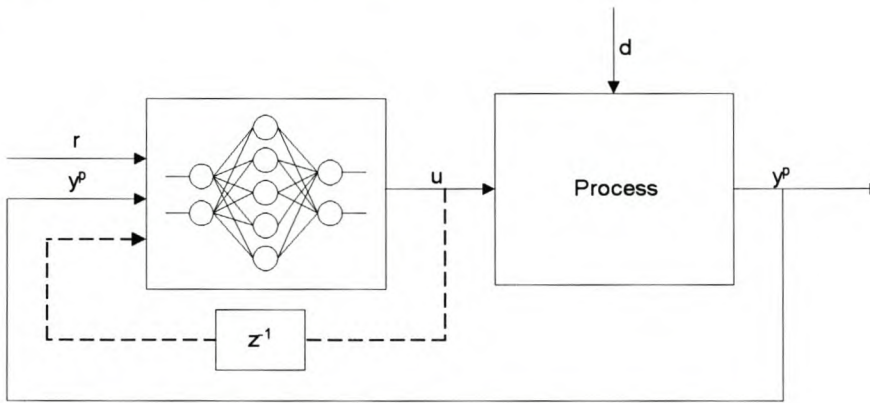


Figure 4-1 - Direct Inverse Control

4.1.3 Model Reference Control

The desired performance of the closed-loop system is specified through a stable reference model M , which is defined by its input-output pair $\{r(t), y^r(t)\}$. The control system attempts to direct the plant output $y^p(t)$ to match the reference model output asymptotically, i.e.

$$\lim_{t \rightarrow \infty} \|y^r(t) - y^p(t)\| \leq \varepsilon \quad (4-1)$$

for some specified constant $\varepsilon \geq 0$. In the connectionist model structure the error, ε , is used to train the *network* acting as the controller. The training procedure will tune the controller as a "detuned" inverse, as defined by the reference model. The control loop structure is illustrated in figure 4-2.

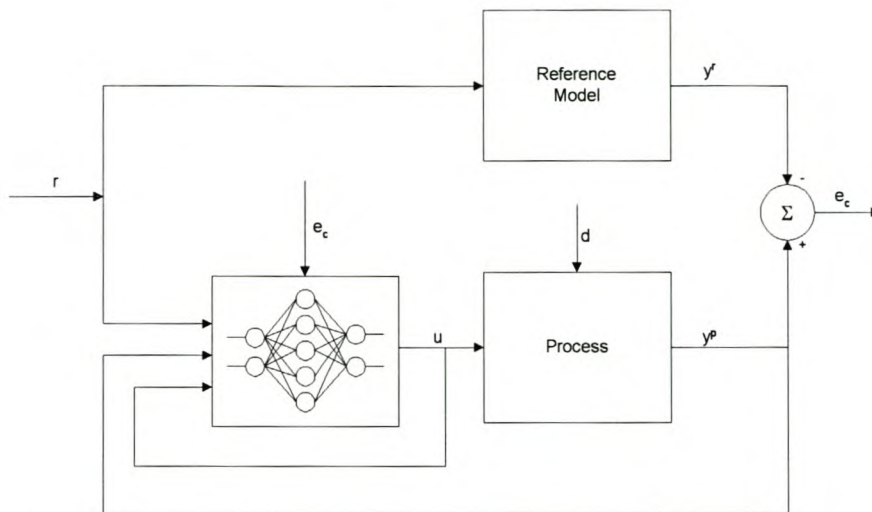


Figure 4-2 - Direct model reference control

4.1.4 Internal Model Control (IMC)

In this structure the system forward and inverse models are used directly as elements within the feedback loop. A system model is placed in parallel with the real system. The difference between the system and model outputs is used for feedback purposes. The feedback signal is processed by a controller subsystem in the forward path. The controller subsystem is related to the system inverse as dictated by the *IMC* properties. Given neural *network* models of the system forward and inverse dynamics, the realisation of *IMC* is illustrated in figure 4-3. A linear filter is usually introduced which may be designed to introduce provide desired robustness and tracking response to the closed loop system.

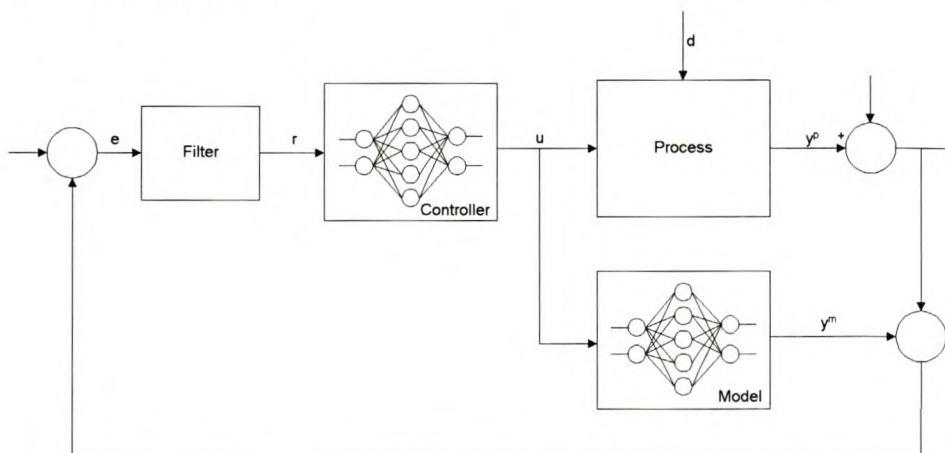


Figure 4-3 - Structure for internal model control

The implementation structure of *IMC* is, however, limited to open-loop stable systems. The technique has, however, been widely applied in process control. As opposed to model predictive control (section 4.1.5), it only predicts one sampling time into the future.

The independence of the model identification task (encoded by the training of the NN model) from the control law (inversion of the NN model) may prove problematic, as it may lead to near singular neural *network* regression (Stephanopoulos & Han, 1996).

4.1.5 Model Predictive Control

Chemical processes are typically operated using linear controllers, although these processes may be highly nonlinear. Model predictive controllers have been successfully extended to nonlinear processes. The advantages to nonlinear predictive control include explicit handling of process time-delays, constraints, the ability to handle nonminimum-phase systems and the incorporation of knowledge of the future set point changes (Sistu et al., 1993).

Model predictive control (*MPC*) is defined as a control scheme in which the controller repeatedly determines (optimises) a *manipulated variable* profile, that optimises an open-loop performance on a time interval extending from the current time to the current time plus a prediction horizon. Feedback is incorporated by using process measurement to update the optimisation problem for the next time step. The receding horizon technique is introduced as a natural, computationally feasible feedback law. The method has proven to have desirable stability properties for nonlinear systems. Also, the generality of the performance objective, as opposed to standard integral square error between output and set point, provides the opportunity to design *MPC* controllers for higher level functions such as energy or waste minimisation (Eaton & Rawlings, 1992).

As a consequence of its structure (figure 4-4), a *MPC* is a feedforward controller for known process changes and a feedback controller for unknown process changes. Thus, *MPC* can reject *measured* disturbances more rapidly than conventional

controllers, by anticipating their impact on the process. Set point changes are achieved efficiently through their ability to predict an optimal sequence of manipulated input values to be implemented. The feedback element of a *MPC* compensates for the effects of *unmeasured* disturbances on the process outputs and deviations between model outputs and those measured (*process/model mismatch*) (Bregel & Seider, 1989).

The prediction horizon allows the *MPC* controller to take control action at the current time in response to forecast error even though the error at the current time is zero. Also the predictive controller may be given information about future constraints and future inputs such as planned set point changes or forecasts of loads or disturbances. Eaton & Rawlings (1992) show that it is precisely this property of the *MPC* controller that is beneficial for controlling (scheduling) *nonminimum phase* plants.

Implementing *MPC* with a neural *network* approach, involves utilising a neural *network* model to provide predictions of the future plant response over the specified horizon. The predictions supplied by the neural *network* are passed to a numerical optimisation routine, which attempts to minimise a specified performance criterion in the calculation of a suitable control signal. The control signal may be chosen so as to minimise a quadratic performance criterion -

$$J = \sum_{j=N_1}^{N_2} [y^r(t+j) - y^m(t+j)]^2 + \sum_{j=1}^{N_2} \lambda_j [u'(t+j-1) - u'(t+j-2)]^2 \quad (4-2)$$

subject to the constraints of the dynamic model. The constants N_1 and N_2 define the horizons over which the tracking error and control increments are considered. The values of λ are the control weights. The remaining parameters are illustrated in figure 4-4 (Hunt et al., 1992).

Another alternative, is to train a further neural *network* to mimic the action of the optimisation routine. The controller *network* is consequently trained to produce the same control output for a given plant output (Hunt et al., 1992).

Model predictive control's multi-step strategy has proven performance in controlling processes in unstable operating regimes. However, the *MPC* approach remains sensitive to modelling errors in these unstable regions. A disadvantage of the *MPC* approach lies in that the execution of the optimisation algorithm is computationally intensive, especially where linearisation of nonlinear systems is not applicable. Also, the solution of the optimisation problem - therefore the controller behaviour - depends on a number of tuning parameters, such as the weighting coefficients in the objective function, the *convergence* criterion, the scaling of the variables and the magnitude of the velocity bounds (Psichogios & Ungar, 1991).

The costs and effort required to implement an advanced control algorithm, such as *MPC*, include the development of models to describe the process dynamics, the dedication of processing power, and the tuning of more parameters relative to analogue controllers. The implementation of *MPC* is normally only justified for processes that cannot be adequately controlled by using less complex algorithms. These processes typically include the production of chemicals in high purities, chemical reactors with multiple steady *state* and periodic attractors, extraction processes with narrow two- and three- phase regions (e.g. supercritical extraction), distillation towers with temperature and concentration fronts sensitively coupled to the reflux ratio (azeotropic distillation towers), and processes required to operate in the region of many design and operating constraints (Bregel & Seider, 1989).

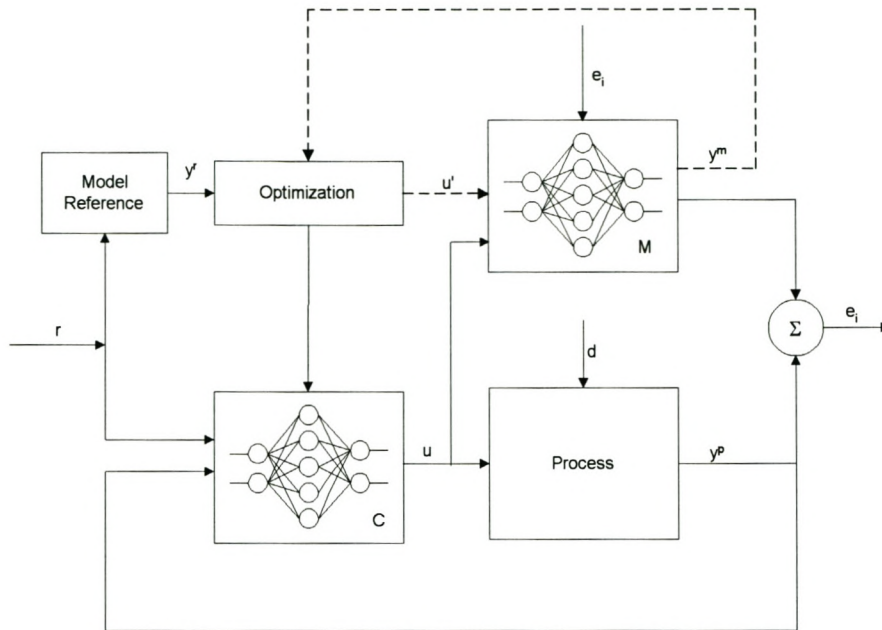


Figure 4-4 - Model Predictive Control structure utilising neural networks

4.2 CONCLUDING REMARKS FOR CHAPTER 4

This chapter describes the numerous control structures in which neural networks have found application. Also, it provides a framework in which to assess the *neurocontrollers* that are developed using the *SANE* algorithm.

4.3 SYMBOLS FOR CHAPTER 4

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
d	Process disturbance	-
e	Error signal	-
r	Set point input	-
u	Controller control action	-
y	<i>Process variable</i>	-
z^{-1}	Z - transform	-

Greek symbols

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
ε	Error constant	-

Superscript

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
p	Process	-
r	Reference model	-
m	Plant model	-

Subscript

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
t	Time	-

5 PROCESS CONTROL NEUROCONTROL SIMULATION CASE STUDIES

OBJECTIVES OF CHAPTER 5

- Elaborate on particular instances of overdesign in the process industries
 - Present *SANE* developed *neurocontrollers* for 4 complex process *domains*, frequently considered in the control literature.
 - Compare the *SANE* controller performance to other considered control strategies, such as model predictive control.
 - Demonstrate *SANE*'s ability to develop *neurocontrollers* that are able to generalise over a wide range of initial process conditions and obtain set points with minimal *state*.
 - Demonstrate *SANE*'s ability to reject unmeasured process disturbances and sensor noise.
 - Demonstrate *SANE*'s ability to compensate for model parameter uncertainties.
 - Demonstrate *SANE*'s ability to integrate process design and optimal control analysis, by searching the *state* space for the optimum economic operating point and developing an optimal controller simultaneously.
-

The following sections (section 5.1 to 5.5) describe four complex process control benchmarks; a bioreactor, a catalytic reactor, a nonisothermal reactor and a Van de Vusse reactor. Each section consists of a complete process model description, a summary of previously considered control implementations for the particular control problem, and the *SANE* developed *neurocontroller*'s implementation.

5.1 NEURAL PROCESS CONTROL UTILIZING SANE

5.1.1 SANE fitness function implementation

Oscillation in the closed loop response to a set point or load change (disturbance), is generally undesirable in chemical process control. Although more underdamped (oscillatory) closed loop systems may present faster settling times, significant overshoot and large decay ratios are typically undesirable approaches to the set point.

An overdamped response (no oscillation) is also undesirable as the long settling times present an unacceptable approach to the set point.

The *ITAE* (integral time absolute error) prototype design transient response characteristics were deemed most appropriate for a wide range of process control problems. The *ITAE* responses are characterised by some overshoot, with a small or insignificant decay ratio. The slight overshoot allows for a faster response (settling time) and the small decay ratio results in almost no oscillation. The *ITAE* responses minimise the integral of the time multiplied by the absolute value of the error (deviation from set point) (equation 5-1), that is, it penalises errors that persist for long periods of time.

$$J = \int_0^{\infty} t \cdot |e| \cdot dt \quad (5-1)$$

The *SANE fitness* function for all the control case studies presented in sections 5.2, 5.3, 5.4 and 5.5, is based on the *ITAE* response criteria. The *SANE* algorithm is required to minimise J (equation 5-1) for each trial *evaluation* of a particular *agent*. The *ITAE* criteria allows for large control actions during the initial stage of the *evaluation* trial, as the first terms in the numerical integration are relatively small (t is small) for large *process variable* deviations from set point. As t grows larger, the set point needs to be reached with minimum *state*, as even a small error has a significant contribution to the minimisation of J in the latter stage of the *evaluation* (equation 5-1).

5.1.2 SANE Control Structure Implementation

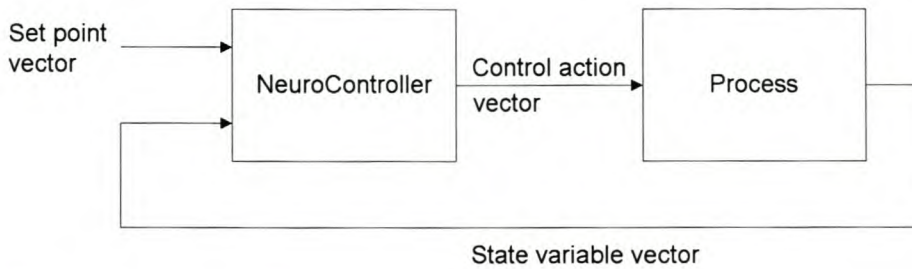


Figure 5-1 - Final Control structure for SANE neurocontroller

The *neurocontrollers* developed with *SANE* (described section 3.6) are incorporated into the closed loop utilising the direct approach (section 4.1). No process model structure is incorporated into the control loop, as with *Internal Model Control (IMC)* (section 4.1.4) or *Model Predictive Control (MPC)* (section 4.1.5). The implementation is, however, not equivalent to direct inverse control (section 4.1.2) as the direct inverse of the process (which represents "perfect control") is not sought.

The *SANE* algorithm is required to search for a behaviour (control) *policy* that possess characteristics of the *ITAE* response (section 5.1.1). In this sense, *SANE* evolves *neurocontrollers* in a manner, which resembles the training in the direct model reference control structure (section 4.1.3). The developed *SANE neurocontroller* may thus be regarded as a "detuned" direct inverse control implementation.

In the *SANE* implementation the *neurocontroller* is, however, not explicitly required to match an explicit reference model. In model reference control structures the output of the plant is required to exactly match the output of the reference model (figure 4-2) to maintain a zero error at each time step. The approach is thus quantitative.

The reliance of *SANE* on a reference model is more implicitly in its use of the *ITAE* criteria in the *fitness* function. The *neurocontrollers* are developed by minimising the *ITAE* function (equation 6-4) in each *evaluation*. No exact mapping of $\frac{y_p}{r}$ dictates the desired response in terms of, for example, damping factors or time constants as would be required in the model reference control structure. The *SANE* approach is

thus qualitative and focused on obtaining a general behaviour characteristic, rather than matching a desired response exactly.

5.2 A BENCHMARK BIOREACTOR CONTROL PROBLEM

The Agrawal bioreactor has been proposed as a chemical process control benchmark, due to the process' complex dynamics (Ungar, 1991). Section 5.2.1 details a process description. Section 5.2.2 to section 5.2.8 presents an overview of control strategies previously considered for the bioreactor. Section 5.2.9 describes the control strategy developed using *SANE*.

5.2.1 Bioreactor Process Description

Agrawal et al. (1982) modelled the behaviour of microbial cultures within the framework of lumped kinetic models, which has proved useful in industrial applications. Biological systems have a penchant for periodicity, with this oscillatory phenomena being verified experimentally in many continuous cultures. These oscillations are assumed to arise from feedback interactions between the cell metabolism and *environment* and interactions among linked intracellular reactions (Agrawal et al., 1982).

The growth rate of most cells in fermentation processes is inhibited by large substrate concentrations. Agrawal et al. (1982) investigated the effect of this inhibition on steady-state and periodic solutions, utilising the following one-hump growth model.

$$\mu\{S\} = k \cdot S \cdot e^{(-S/K)} \quad (5-2)$$

where μ is the specific growth rate (maximum at $S = K$) and S is the substrate concentration. The specific substrate consumption rate, σ , is related to the specific growth rate through the yield coefficient, Y .

$$Y\{S\} = \frac{\mu\{S\}}{\sigma\{S\}} = a + b \cdot S \quad (5-3)$$

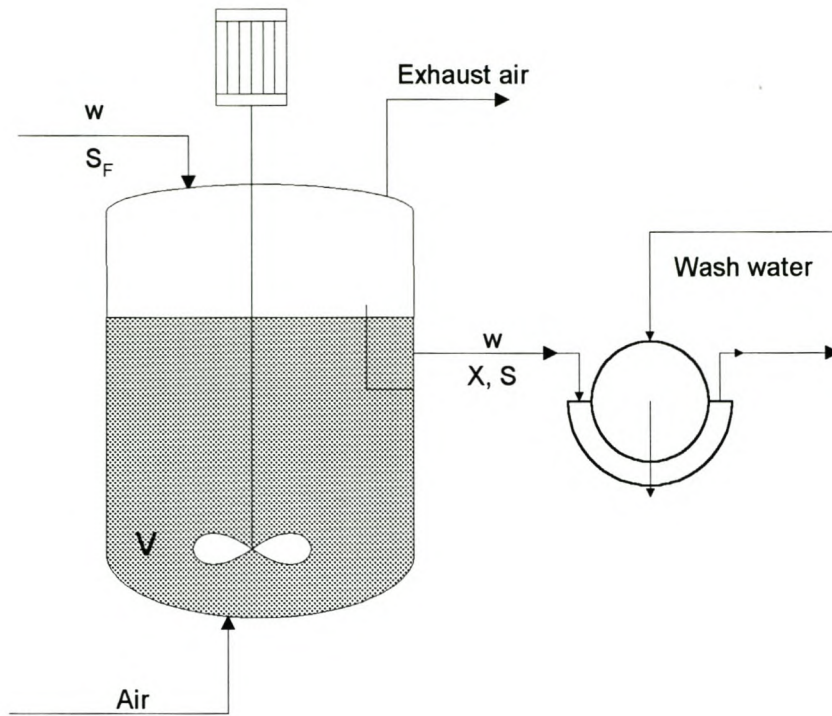


Figure 5-2 - Schematic diagram of the bioreactor

For the CSTR in figure 5-2, the cell and substrate mass balances are described by the following coupled nonlinear differential equations.

$$\frac{dX}{dt} = -\frac{F}{V} \cdot X + \mu\{S\} \cdot X \quad (5-4)$$

$$\frac{dS}{dt} = \frac{F}{V} \cdot (S_F - S) - \sigma\{S\} \cdot X \quad (5-5)$$

Dedimensionalising, these mass balances yield equation 5-6 and equation 5-7, with the model parameters definitions in table 5-1 (Bregel & Seider, 1989).

$$\frac{dC_1}{d\tau} = -C_1 \cdot w + C_1 \cdot (1 - C_2) \cdot e^{(C_2/\gamma)} \quad (5-6)$$

$$\frac{dC_2}{d\tau} = -C_2 \cdot w + C_1 \cdot (1 - C_2) \cdot e^{(C_2/\gamma)} \cdot \frac{(1 + \beta)}{(1 + \beta - C_2)} \tag{5-7}$$

Table 5-1 - Model parameter formulation for the Agrawal bioreactor

<i>Description</i>	<i>Formulation</i>	<i>Unit</i>
Dimensionless Cell Mass	$C_1 = \frac{X}{(S_F \cdot Y\{S_F\})}$	[]
Dimensionless Substrate Conversion	$C_2 = \frac{(S_F - S)}{S_F}$	[]
Weighting coefficient	$\beta = \frac{a}{(b \cdot S_F)}$	[]
Weighting coefficient	$\gamma = \frac{K}{S_F}$	[]
Dimensionless time	$\tau = t \cdot \left(\frac{F}{V}\right)$	[]
Flow rate	$w = \frac{1}{Da}$	[]
Damkohler number - ratio of specific growth rate to the residence time	$Da = \frac{\mu\{S_F\}}{(F/V)}$	[]

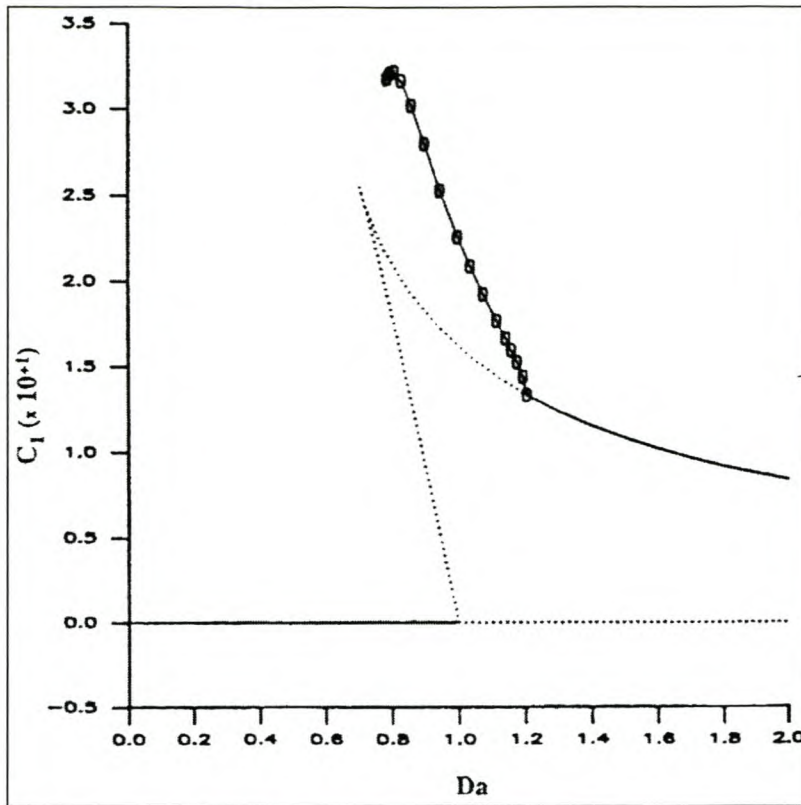


Figure 5-3 - Bioreactor Multiple Steady States - Stable (—), Unstable (····) and Periodic (---) steady states of cell mass as a function of Damkohler number for the Agrawal bioreactor with $\gamma = 0.48$ [] and $\beta = 0.02$ [] (Bregel & Seider, 1989).

To identify the regimes of hysteresis and periodic operation, Agrawal et al. (1982) varied the parameters Da , β , and γ . Figure 5-3 demonstrates the nonlinear steady-state and periodic branches of the solution diagram with $\gamma = 0.48$ [] and $\beta = 0.02$ []. The stable steady state that yields the highest cell mass occurs at the Hopf bifurcation point, where $Da = 1.206$ []. Beyond that, the steady state attractors are unstable and the periodic attractors are stable with large periods. These observations correspond to the experimental observations for the aerobic fermentation of *Saccharomyces cerevisiae* on a sugar cane - molasses medium. Experimental verification has demonstrated the existence of periodic regimes in which the cell mass builds until limited by the available oxygen and is consequently depleted until the oxygen becomes available in excess (Bregel & Seider, 1989).

Fermentation control presents a difficult control problem, as the dynamic behaviour of such fermenters is invariably non-linear and model parameters vary in an unpredictable manner. Accurate process models are rarely available due to the

complexity of the underlying biochemical processes. Also, a lack of reliable biosensors make the process *state* difficult to characterise. Although the control of fed-batch fermenters has been widely researched, continuous operation becomes desirable when the production of biomass or product is to be optimised. Productivity optimisation for continuous fermenters has typically incorporated the use of open-loop control policies. Once the optimum has been determined (generally open loop stable), a feedback controller is required to reject disturbances and to account for time-varying behaviour (Henson & Seborg, 1992).

Several linear feedback strategies, which include proportional and proportional-integral control, as well as adaptive control have been proposed. Several of these approaches have neglected the nonlinearities inherent in most fermentation models (Henson & Seborg, 1992).

5.2.2 Bioreactor Control with a Multi-step Nonlinear Predictive Controller

As concluded by Brengel & Seider (1989), in the design stage, it may seem prudent to operate at the highest yields of cell mass, but not too closely to the region of periodic oscillations. A designer might select $Da = 1.30$ [] and examine the controllability of the process in response to changes in the setpoint (C_1^{SP}) and unmeasured disturbances. At $Da = 1.3$ [], the reactor is marginally open loop stable (figure 5-1). Although simple PID controllers can be tuned to stabilise these steady states, Brengel & Seider (1989) reported PID controller responses that were far more sluggish and oscillatory.

In order to explore the controllability in the region of such a steady state, Brengel & Seider (1989) introduced deviation variables, $X_1(t) = C_1(t) - C_{1s}$ and $X_2(t) = C_2(t) - C_{2s}$. A setpoint change was introduced that increases the dimensionless cell mass by 0.05 [] or $C_1 = 0.1707$ [] to demonstrate the servo characteristics (set point change response) of the Multi-step Nonlinear Predictive Controller (MNPC). For the uncontrolled process, figure 5-3 indicates that this new setpoint exhibits multiplicity with a periodic attractor and two unstable steady-state attractors. A sampling time of $T = 0.5$ [] was used, based upon typical transient responses in the vicinity of the initial steady state. The best controlled response for this set point change is illustrated

in figure 5-4, which was achieved after performing a sensitivity analysis by varying the weights on the *state* variables and prediction horizons.

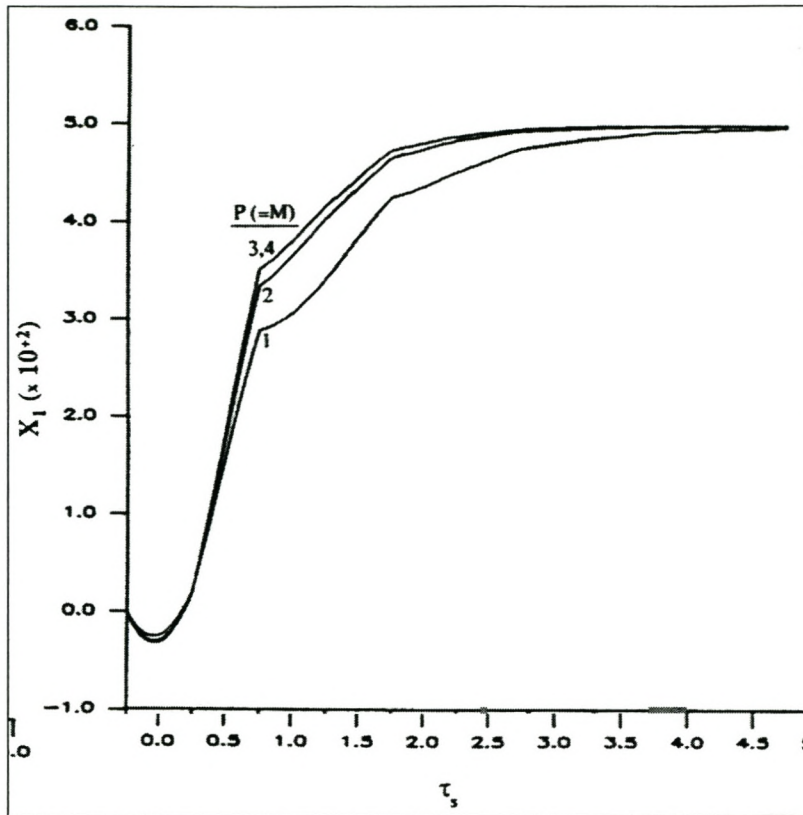


Figure 5-4 - Transient response to step change in C_1^{SP} with a prediction horizon of $P = 4$. X_1 represents a deviation variable from the nominal set point ($C_1^{SP} = 0.1707$ []) (Bregel & Seider, 1989).

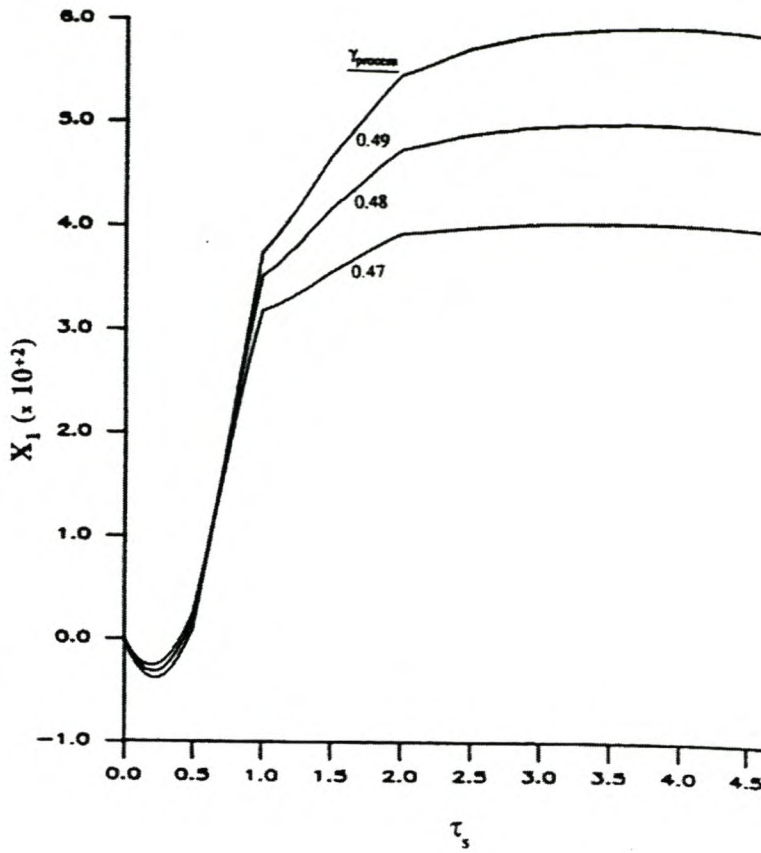


Figure 5-5 - Transient response to step change in C_1^{SP} with model mismatch in γ . (Bregel & Seider, 1989)

Figure 5-5 below illustrates the impact of model errors ($\leq 2\%$) in the model parameter γ when the *MPC* is not augmented with a mismatch compensation algorithm, which adjusts the model parameters online to minimise the *state*. In figure 5-5, γ_{model} is held constant at $\gamma_{\text{model}} = 0.48$ [] and the process is assumed to operate with an effective γ_{process} ranging from $\gamma_{\text{process}} = 0.47 - 0.49$ []. Without *model mismatch compensation*, significant offset results in the steady state should *process/model mismatches* occur. In the event of $\gamma_{\text{process}} \neq \gamma_{\text{model}}$, γ_{model} and Da_s may be appropriately estimated (adjusted) to eliminate the offset as illustrated in figure 5-6. This process mismatch compensation is possible, as the steady state formulation of equation 5-6 and equation 5-7 may be satisfied by a single w , when $X_1 = X_1^{SP}$ and $X_2 = X_2^{SP}$, as γ_{model} and Da_s vary (β_{model} fixed) (Bregel & Seider, 1989).

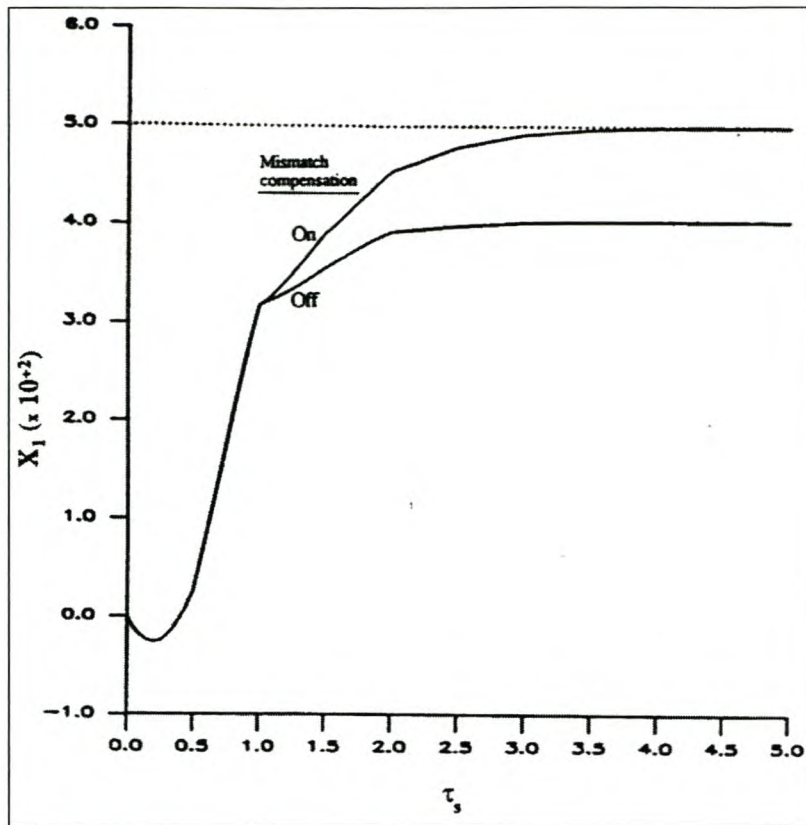


Figure 5-6 - Transient response to step change in C_1^{SP} with process / mismatch. The steady state correction through mismatch compensation is illustrated (Bregel & Seider, 1989).

Similar *state* is evident for model errors in the order of 20% in the model parameter β . The offset cannot be eliminated by *model mismatch compensation* as the two *process variables* (X_1 and X_2) are adjusted with a single *manipulated variable* (w). Should $\beta_{\text{process}} \neq \beta_{\text{model}}$, mutually exclusive values of w are obtained, should $X_1 = X_1^{SP}$ and $X_2 = X_2^{SP}$ be substituted into equation 5-6 and equation 5-7 respectively. Consequently, the offset cannot be eliminated using a *model mismatch compensation* technique. To eliminate the offset with *model mismatch compensation* a second feed stream, of a different concentration, may be added to provide a second *manipulated variable* (Bregel & Seider, 1989).

5.2.3 Bioreactor control with Kalman Filter Trained Recurrent Networks

Puskorius & Feldkamp (1994) also considered the process described in section 5.2.1. A recurrent neural *network* with two hidden layers of 15 and 10 sigmoidal nodes was used to model the input/output behaviour of the bioreactor for the full-*state* (all the *state* variables are available for control purposes) problem. The inputs to the neural

network process model at time step k , are the cell mass $C_1(k)$ and the nutrient conversion $C_2(k)$, along with the control action $w(k)$. The output layer consists of two linear summing nodes; the outputs corresponding to predictions of the *states* at time $k+1$. The identification neural *network* was trained with a fixed sequence of 10 000 data points, which was generated by simulation of the dynamic model equations 5-6 and equation 5-7. The resulting neural *network* contained 130 trainable *weights* (Puskorius & Feldkamp, 1994).

Given the pretrained identification *network*, a controller *network* was developed. Puskorius & Feldkamp (1994) anticipated the need for a degree of retraining for the identification *network*, as training of the controller *network* proceeded into the operating region characterised by unstable steady *states*. The original identification *network* was only trained with control signals that did not explore these regions. No retraining was, however, necessary as the identification *network* appeared to be capable of providing reasonably good estimates of the dynamic derivatives as training proceeded into the unstable region.

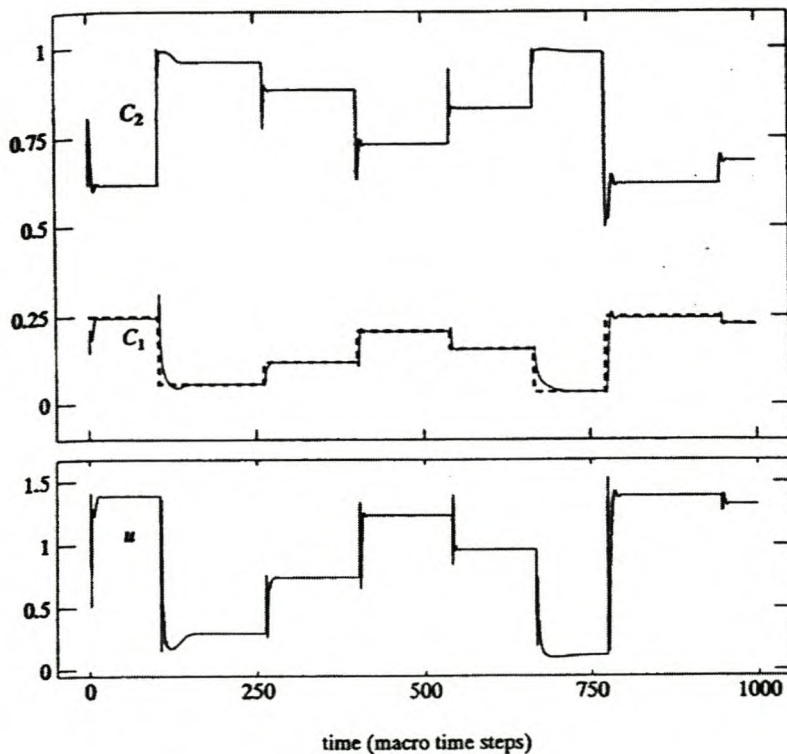


Figure 5-7 - Neural network model reference control for the nominal plant parameters with full-state information. The set point is plotted as a dashed line (Puskorius & Feldkamp, 1994).

Upon completion of training the overall performance of the final controller *network* is illustrated in figure 5-7. Puskorius and Feldkamp (1994) also tested the robustness of the controller by applying it (without retraining) to a plant with altered parameters ($\gamma = 0.456$ and $\beta = 0.016$). The parameter alterations correspond to a 5% deviation in γ and a 20% deviation in β . *Gaussian* measurement noise of standard deviation equal to 0.01 was added to both *state* variables. Representative results are shown in figure 5-8.

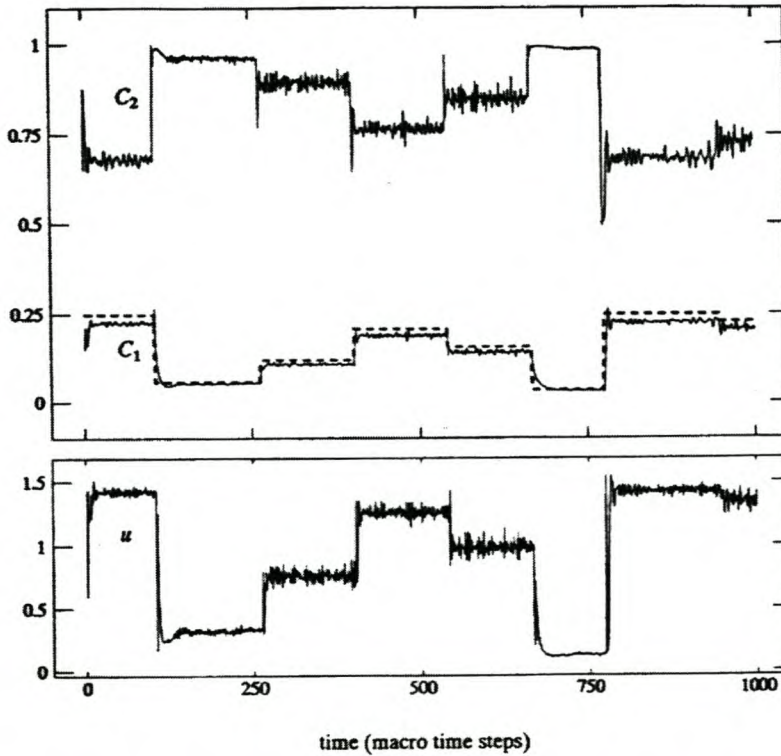


Figure 5-8 - Performance of the controller network for a bioreactor characterised by a 5% change in γ and 20% in β and added measurement noise. The dashed lines represent the reference signal; the measurement noise reaching the system through the controller (Puskorius & Feldkamp, 1994).

5.2.4 Neurocontrol using a simplification of the Backpropagation-Through-Time-Algorithm

Backpropagation-through-time (BPTT) is an extension of backpropagation which allows a multi-layer neural *network* to approximate an optimal control law. Some prior knowledge (Jacobian matrixes) of the process is required. Bersini & Gorrini (1997) implemented the nominal benchmark specification (section 5.2.1), except that the sampling period was $\frac{1}{50}$ that of the standard sample period. This was required as

the BPTT method requires that the partial derivatives are available in order to evaluate the Jacobians, and these derivative are difficult to calculate over the standard sample period. They attempted to train a neural *network* with 10 *neurons* in the hidden layer (Bersini & Gorrini, 1997).

"It turned out to be quite difficult to learn a control strategy that is valid over a wide range of different initial conditions. We performed several tests using either 10 or 20 neurons in the hidden layer, different learning rates and different control horizons. We tried to optimise around a stable equilibrium point and an unstable one. In all cases the system seems to converge (the square error decreases), but the reactor eventually collapses."(Bersini & Gorrini, 1997)

5.2.5 Bioreactor control utilising functional expansion models

Harris et al. (1998) proposed to control the Agrawal bioreactor (section 5.2.1) in the stable steady state region.

Table 5-2 - Harris et al. (1998) nominal operating steady state for the bioreactor.

<i>State Variable</i>	<i>Steady state Value</i>
$C_{1,s}$	0.0965 []
$C_{2,s}$	0.912 []
w	0.588 []

At this operating point, although in the stable steady *state*, the reactor displays both nonminimum-phase (inverse response to an appropriate *manipulated variable* change) and oscillatory behaviour. As the gain of the process is fairly linear, functional expansion models (FEx) and a linear model capture the steady-state behaviour of the system accurately over a limited range. However, the dynamic of the system are nonlinear. In the open-loop the inverse response of the nonminimum phase behaviour is in evidence (Harris & Palazoglu, 1998).

The FEx models proved to have a limited range of application for this process. None of the models capture the Hopf bifurcation. Also, the FEx models have a limited range of *convergence* as the Damkohler number is reduced. The FEx models capture the

oscillatory behaviour of the system for a positive step to SP($C_1 = 0.1265, C_2 = 0.8720$), while the linear model has significant oscillatory behaviour. In the closed loop, the FEx controller displays only slight oscillations, while the linear controller suffers from sustained oscillation (figure 6.9)

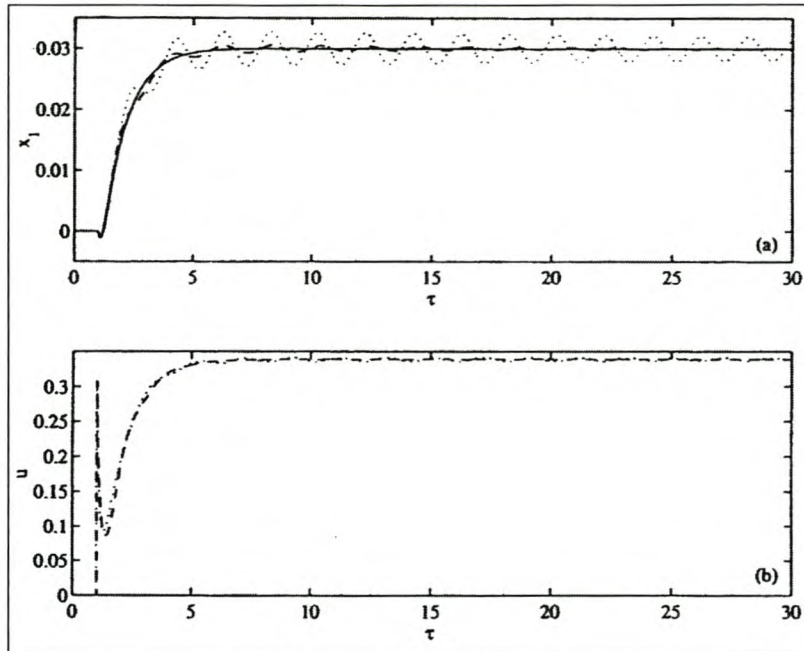


Figure 5-9 - Closed loop behaviour of the bioreactor for a set point change from SP(0.0965, 0.912) to SP(0.1265, 0.8720). x_1 is the cell mass concentration deviation from the initial set point. u is the flow rate change from the initial steady state (\cdots , linear model), ($---$, FEx model), ($—$, numerical) (Harris & Palazoglu, 1998).

However, negative step directions from the nominal set point to SP($C_1 = 0.0785, 0.9345$) forces the system towards the region where the FEx models suffer from a limited range of *convergence*. In the closed loop, the linear controller is robust, while the FEx controller is oscillatory due to the approach to instability of the model (figure 6-10). For reference moves in cell mass concentration greater than $dx_1 > -0.021$ from the nominal steady state, the FEx controller is unstable (Harris & Palazoglu, 1998).

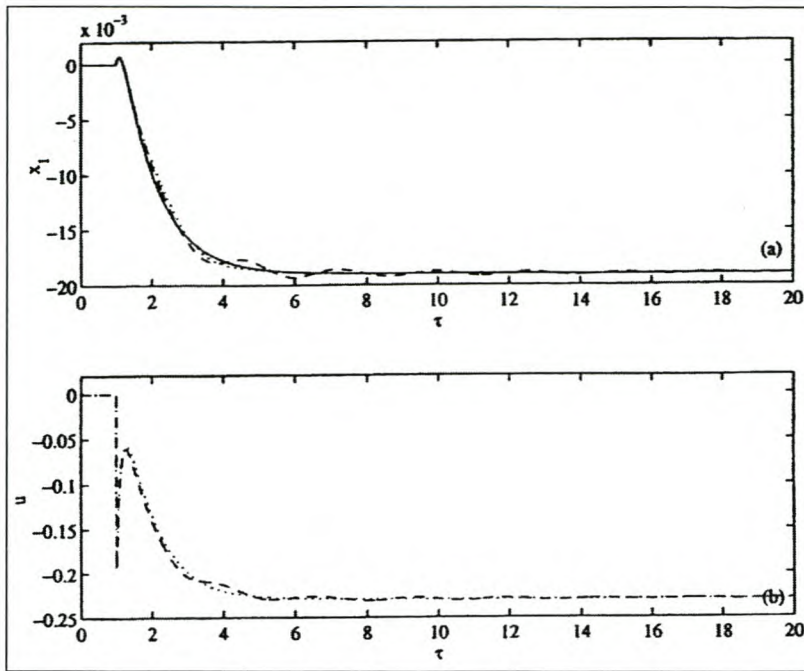


Figure 5-10 - Closed loop behaviour for a set point change from SP(0.0965, 0.912) to SP(0.0785, 0.9345). x_1 is the cell mass concentration deviation from the initial set point. u is the flow rate change from the initial steady state (\cdots , linear model), (---, FEx model), (—, numerical) (Harris & Palazoglu, 1998).

5.2.6 NeuroControl Workbench Results

Jarmulak et al. (1997) modelled the bioreactor identification *network* with a 8-20 20-2 neural *network* configuration. The *network* inputs (8 nodes) consisted of $C_1(k)$, $C_1(k-1)$, $C_1(k-2)$, $C_2(k)$, $C_2(k-1)$, $C_2(k-2)$, $w(k)$ and $w(k-1)$. A training set of 400 patterns, generated randomly over part of the operating range ($C_1[0.0; 0.5]$ and $C_2[0.1, 0.9]$) was used to train the identification *network*. A single input, output pattern was generated by choosing a random plant *state* and running the plant until all the time delay lines (TDLs) were filled. A batch training method, cumulative delta rule, was used to train the identifier *network* off-line for a total of 3000 epochs.

A model-based predictive controller was used to control the bioreactor. A standard minimisation algorithm was used to form the controller of the control system. The controller performance is illustrated in the figure 5-11 below.

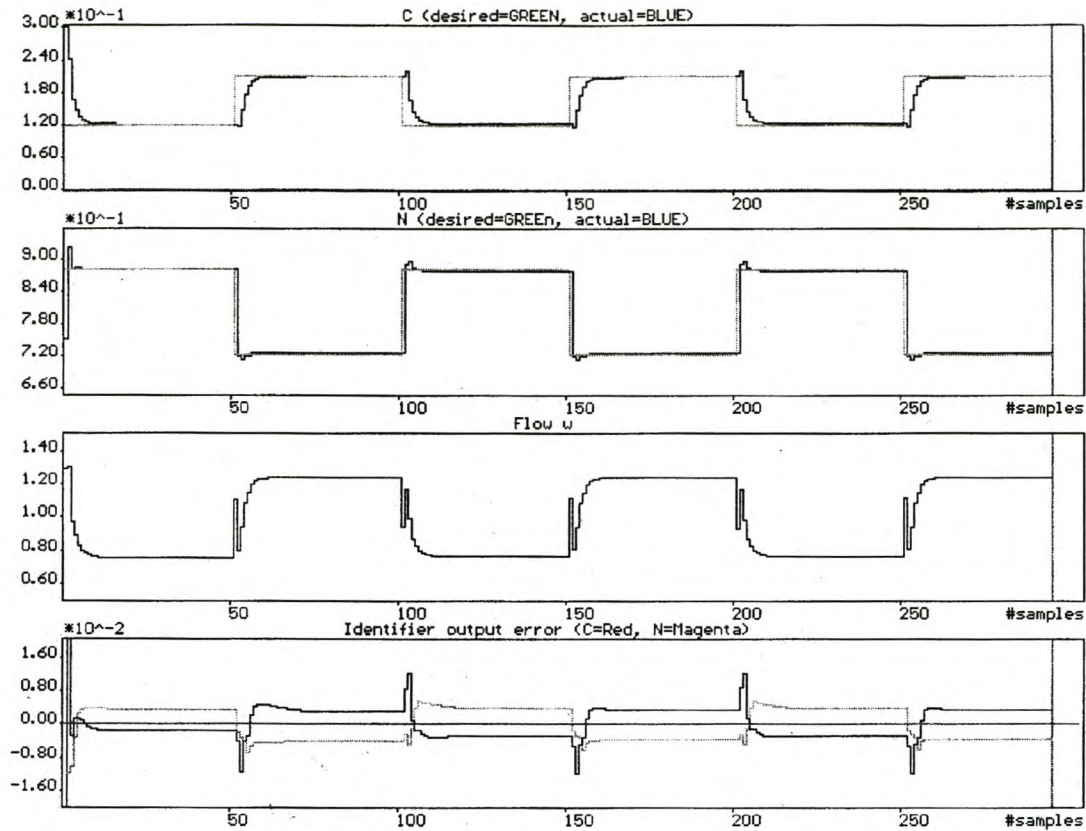


Figure 5-11 - NeuroControl Workbench control system servo response (C_1, C_2), for set point changes between (0.1207, 0.88) and (0.2107, 0.7226) (Jarmulak et al., 1997)

5.2.7 Neural reinforcement control using ELEGANCE

Jarmulak et al. (1997) considered a *reinforcement learning* neurocontrol implementation, in which fully encoded neural *networks* (section 3.4) are evolved. After the *evaluation* of 1500 controllers, a satisfactory *neurocontroller* with 7 hidden nodes was evolved. This implementation differs fundamentally from *SANE*, in that complete reinforcement solutions are encoded, while in *SANE* partial solutions must form complete solutions (section 3.4). The closed loop response is illustrated in figure 5-12.

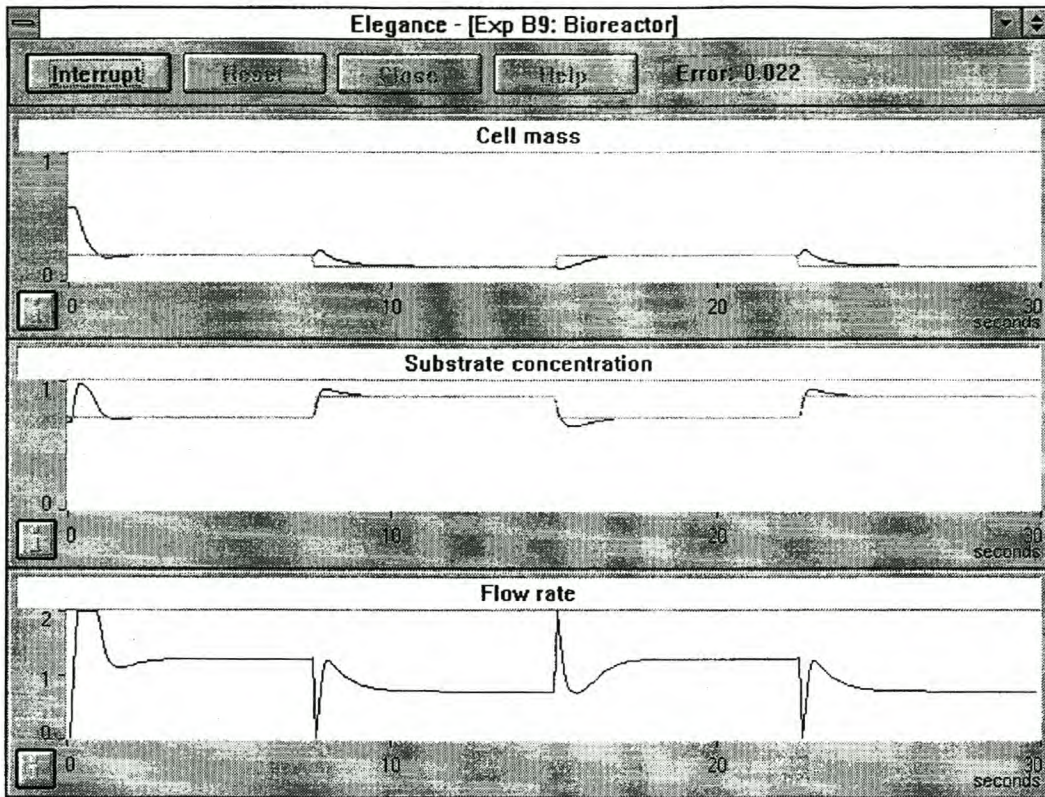


Figure 5-12 - Elegance control system servo response (C_1, C_2), for set point changes between (0.1207, 0.88) and (0.2107, 0.7226) (Jarmulak et al., 1997).

5.2.8 Coevolutionary Bioreactor Reinforcement Learning implementation

Paredis (1997) evaluated the nominal bioreactor control problem (section 5.2.1) described by Ungar (1991) utilising *reinforcement learning* with a *coevolutionary genetic algorithm* (CGA). CGA combines coevolution and life-time *fitness evaluation* (LTFE) to direct the genetic search. CGA utilises full *encoding* of the neurocontrol, but differs from the ELEGANCE implementation (section 5.2.7) in that a *population* of solutions (controllers) and initial conditions (start states) are coevolved in an *evolutionary* (prey-predator) "arms race". Such an arms race involves the step-wise increase in the complexity of both predator and prey. In the described implementation, the start *state population* is, however, not subjected to evolution. Direct interaction between a *neurocontroller* and a start *state* provides *evaluation* for the *fitness* assignment to both the start *state* and the *neurocontroller*. Further interaction between the two *populations* is governed by the respective *fitnesses* of the *individuals* in each

population. Life-time *fitness evaluation* allows for the *individual's fitness* to be determined over a number of generations, which allows for robust *fitness evaluation*.

The initial *population* of start *states* is selected in a randomly selected 10% bandwidth from the desired set point. A *neurocontroller* utilising 12 hidden nodes was evolved after 5000 basic cycles, which involves 100 000 encounters between the start *state* and controller *populations*. The obtained servo responses, however, remain oscillatory around the set point with small amplitudes.

5.2.9 Concluding remarks for control strategies considered in previous research

The bioreactor process control benchmark has been widely used to demonstrate new process control algorithms. The control problem is as a useful benchmark for evaluating control algorithms, due to the complex regions of operation and multiplicity inherent in the process dynamics (section 5.2.1).

The nonlinear model predictive control (NMPC) implementation (section 5.2.2) displayed good servo characteristics, despite changing the operating point from a stable to an unstable plant *state*. The NMPC algorithm, however, proved sensitive to model parameter uncertainties in γ and β . Although *model mismatch compensation* could be used for to eliminate the *state* due to model uncertainty in γ ; mismatch compensation may not be utilized to correct for uncertainties in β . Offset as a result of uncertainties in β , may only be eliminated by introducing an additional substrate feed stream to the reactor. A NMPC implementation may thus require greater capital expenditure to ensure effective control. An *Internal Model Control (IMC)* implementation (section 5.2.3) using neural *networks* for both the process model and the controller, displayed excellent model parameter uncertainty rejection, which may be ascribed to the *generalisation* ability of neural *networks*.

In the two *reinforcement learning* methods (section 5.2.7 & 5.2.8) no *generalisation* analysis is presented. Only the results for set point changes, which the controller was trained to accomplish, are presented in each case. The *neurocontroller* would be

expected to display a high level of performance for areas of the *state* space in which it learned to control the process. No real indication of the *neurocontroller* performance is thus available. The degree of *generalisation* ability afforded the developed *neurocontroller* by the full *encoding reinforcement learning* technique, Elegance, may thus not be assessed. The degree of *generalisation* ability afforded the developed *neurocontroller* by the use of an *evolutionary* "arms race" technique and life-long-*fitness-evaluation*, may thus not be assessed.

5.2.10 Symbiotic, Adaptive Neuro Evolution (*SANE*)

5.2.10.1 Learning optimal behaviour

The *SANE* algorithm was used to evolve a *neurocontroller* with 4 input nodes, 8 hidden nodes (80 connections) and one output node. The four inputs to the input nodes are the cell mass concentration and desired set point, and the substrate conversion (concentration) and desired set point. The single output node determines the *manipulated variable* control action, which is the substrate flow rate to the bioreactor. *SANE* was required to develop the desired control behaviour utilising a maximum of 8 hidden nodes. The final developed *neurocontroller* contained 7 partially connected hidden nodes, as one of the hidden nodes was made redundant by *SANE* by not allotting the redundant node a connection to the output layer. This demonstrates *SANE*'s ability to prune the number of hidden nodes, should the initial number be over specified.

The *neurocontroller* was required to learn to control the process at three set points. The set point at $SP_1(C_1 = 0.1207, C_2 = 0.88)$ is an open loop stable steady *state*, with the other two setpoints at $SP_2(C_1 = 0.2107, C_2 = 0.7226)$ and $SP_3(C_1 = 0.2542, C_2 = 0.5392)$ representing open loop unstable steady *states*. The three steady *state* set points are presented to the *neurocontroller*, over random intervals of ± 50 time units, during learning. For each *evaluation* the trial commences at a random initial condition over the full range of the cell mass concentration and the substrate conversion (concentration). These initial conditions may often be impossible to attain in a real system. The *environment* (simulation) in which *SANE* is required to learn appropriate

control behaviour, is presented without disturbances or noise during each *agent's evaluation*. The final *neurocontroller's* ability to deal with disturbance or sensor noise is consequently a result of effective *generalisation*. The transients for a typical learning trial for the best evolved controller is illustrated in figures 5-13 to figure 5-15. The *neurocontroller's* performance for the learned set point changes is very satisfactory, but not a clear indication of *neurocontroller* performance (*generalisation*) in the presence of uncertainties.

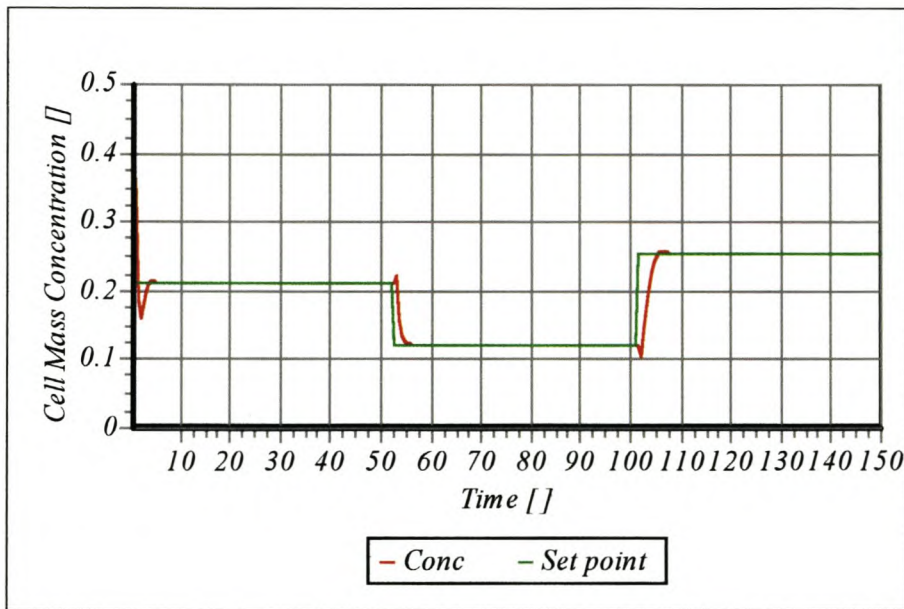


Figure 5-13 - Learning trial transient response for the cell mass concentration for the fittest neurocontroller. SP_2 ($C_1 = 0.2107$, $C_2 = 0.7226$); $SP_1 = (C_1 = 0.1207, C_2 = 0.88)$ & $SP_3(C_1 = 0.2542, C_2 = 0.5392)$. Initial condition $C_1 = 0.33$ and $C_2 = 0.36$.

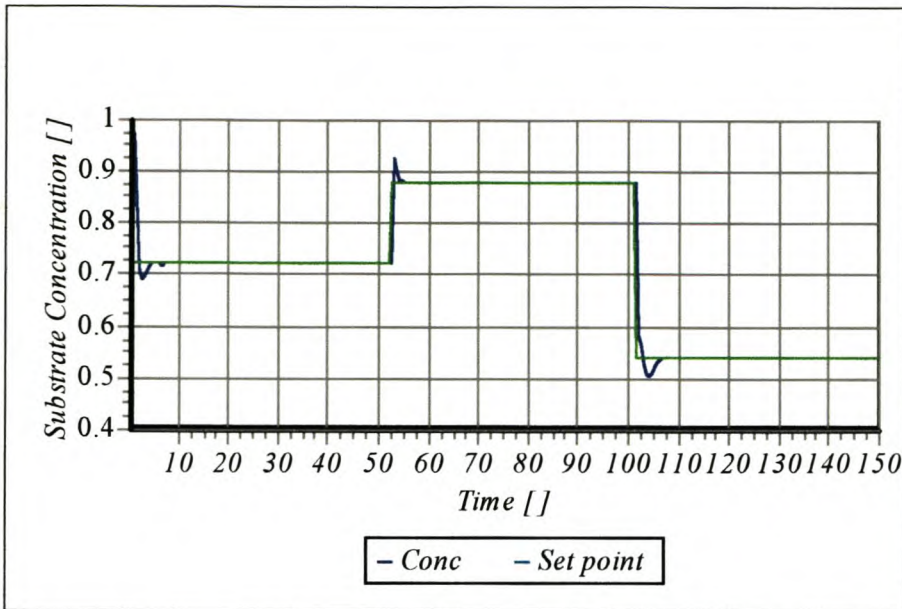


Figure 5-14 - Learning trial transient response for the substrate conversion for the fittest neurocontroller. SP_2 ($C_1 = 0.2107$, $C_2 = 0.7226$); $SP_1 = (C_1 = 0.1207, C_2 = 0.88)$ & $SP_3(C_1 = 0.2542, C_2 = 0.5392)$. Initial condition $C_1 = 0.33$ and $C_2 = 0.36$.

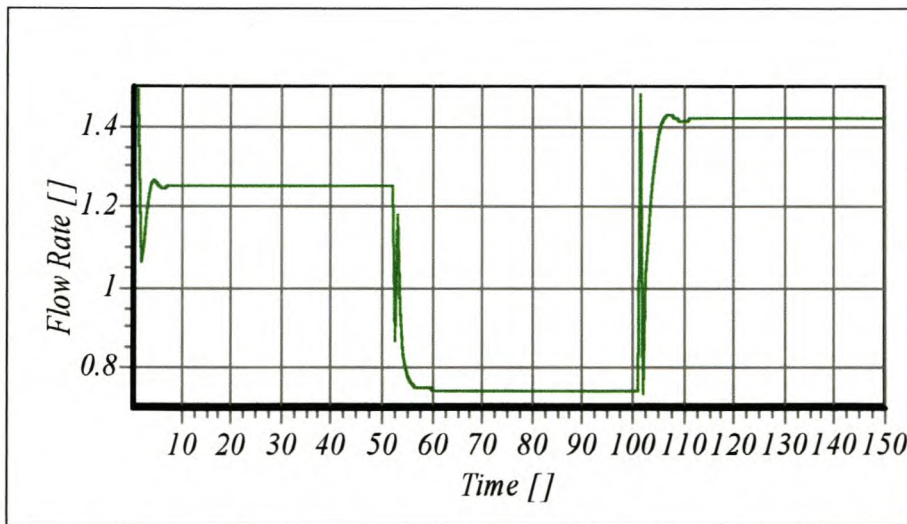


Figure 5-15 - Learning trial flow rate control action for the fittest neurocontroller. SP_2 ($C_1 = 0.2107$, $C_2 = 0.7226$); $SP_1 = (C_1 = 0.1207, C_2 = 0.88)$ & $SP_3(C_1 = 0.2542, C_2 = 0.5392)$. Initial condition $C_1 = 0.33$ and $C_2 = 0.36$.

5.2.10.2 Rejecting Nonminimum Phase Behaviour

The transient response of the *state* variables for a learning trial is also illustrated in figure 5-16 to figure 5-18 for the set point change from a stable to an unstable steady *state*. The non-minimum phase behaviour is evident in the cell mass concentration response (figure 5-16). Nonminimum phase behaviour is characterised by a *process*

variable initially having an inverse response to an appropriate *manipulated variable* control action. This inverse response usually arises from competing dynamic effects that operate on two different time scales.

Nonminimum phase behaviour may cause an excessively oscillatory response should a conventional linear controllers be utilised. This results as the error signal due to the new set point initially increases, even though the *manipulated variable* has changed in the appropriate direction. This further increase in the error signal causes the linear controller to overstate the required *manipulated variable* control action, often resulting in significant overshoot, oscillatory decay and long settling times. In the presence of significant continued disturbances this may result in instability of the closed loop response.

From the *manipulated variable* (flow rate) control action in figure 5-18, it is apparent that the *neurocontroller* has learned to avoid causing an oscillatory response to a set point change, by anticipating or counteracting the effect of the inverse response. An initial high flow rate is induced on set point change, which causes the inverse response, after which the flow rate is returned to its initial rate to disallow the inverse response, after which the flow rate is gradually increase (figure 5-18). The cell mass concentration and the substrate concentration (conversion) transients conform to the *ITAE* response criteria, which allows for slight overshoot, but no oscillation in the decay to set point.

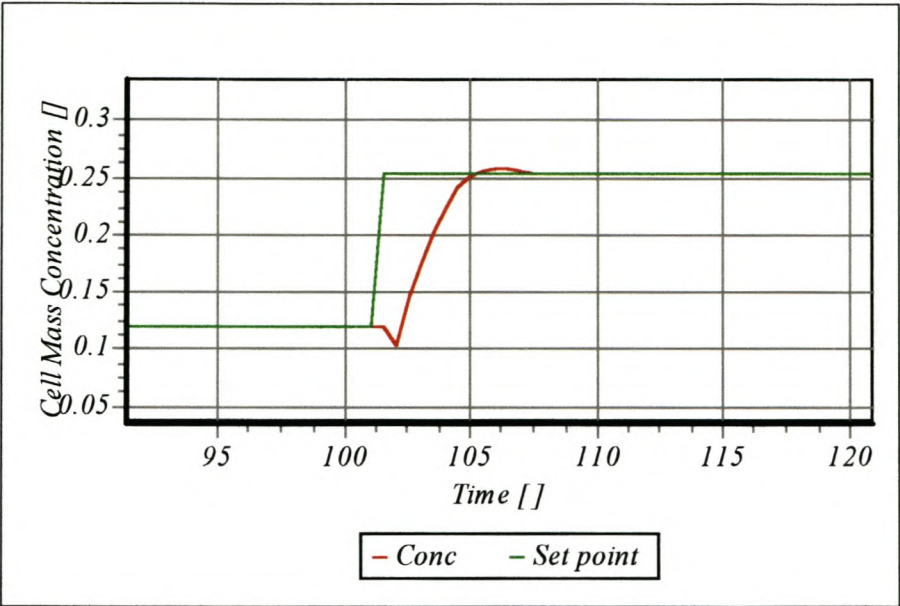


Figure 5-16 - Set point change (learned behaviour) cell mass concentration response from stable $SP_1 = (C_1 = 0.1207, C_2 = 0.88)$ to unstable $SP_3(C_1 = 0.2542, C_2 = 0.5392)$ region of the state space.

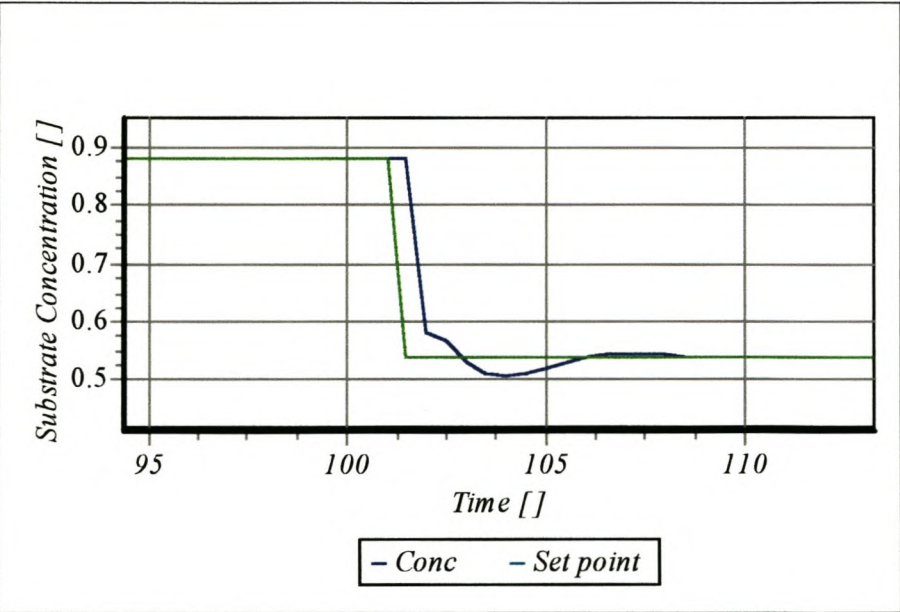


Figure 5-17 - Set point change (learned behaviour) substrate conversion response from stable $SP_1 = (C_1 = 0.1207, C_2 = 0.88)$ to unstable $SP_3(C_1 = 0.2542, C_2 = 0.5392)$ region of the state space.

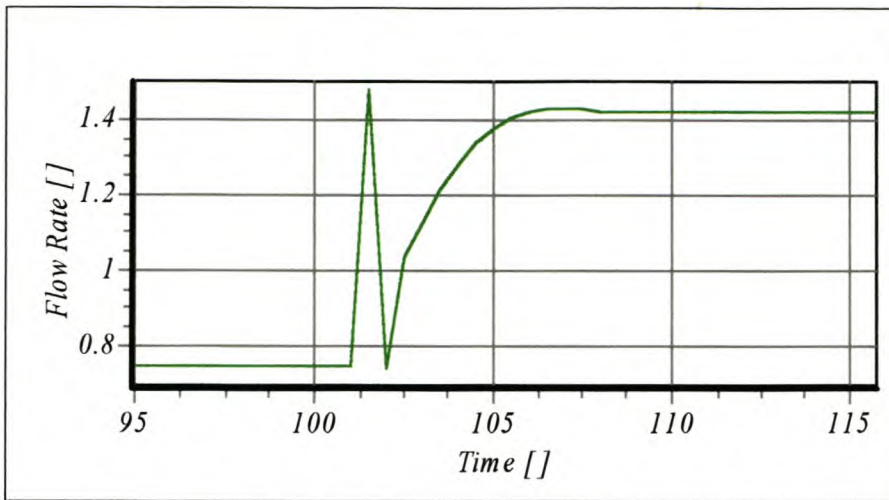


Figure 5-18 - Set point change (learned behaviour) control action from stable $SP_1 = (C_1 = 0.1207, C_2 = 0.88)$ to unstable $SP_3 (C_1 = 0.2542, C_2 = 0.5392)$ region of the state space.

5.2.10.3 Improving the dynamic response of the system

As indicated by Brengel & Seider (1989) a conservative process design may limit the bioreactors operation to the stable steady state region of the bifurcation analysis (figure 5-3). Harris et al. (1998) also proposed developing a controller for operation in the stable steady state region. Although this may result in simplified operation and control of the bioreactor, a significant economic penalty is incurred by avoiding operation at unstable steady states (section 1.3.2). In order to demonstrate the ability of *SANE* to develop *neurocontrollers* that significantly improve the open loop response dynamics, operation in the open loop is considered.

In the open loop stable steady state region the open loop response remains oscillatory as described in figure 5-19 and figure 5-20. As indicated by Henson and Seborg (1992) productivity optimisation of continuous fermenters typically involves open loop control policies, with feedback introduced to reject disturbances at these open loop stable "optimum" operating regions. For the oscillatory long decay time illustrated, a further control objective would be to improve the dynamics of the system by introducing feedback. The *neurocontroller's* ability to improve the dynamics around such a stable steady state point is illustrated in figure 5-21 to figure 5-23. The oscillatory nature of the open loop response is suppressed and the settling time is reduced by an order of magnitude.

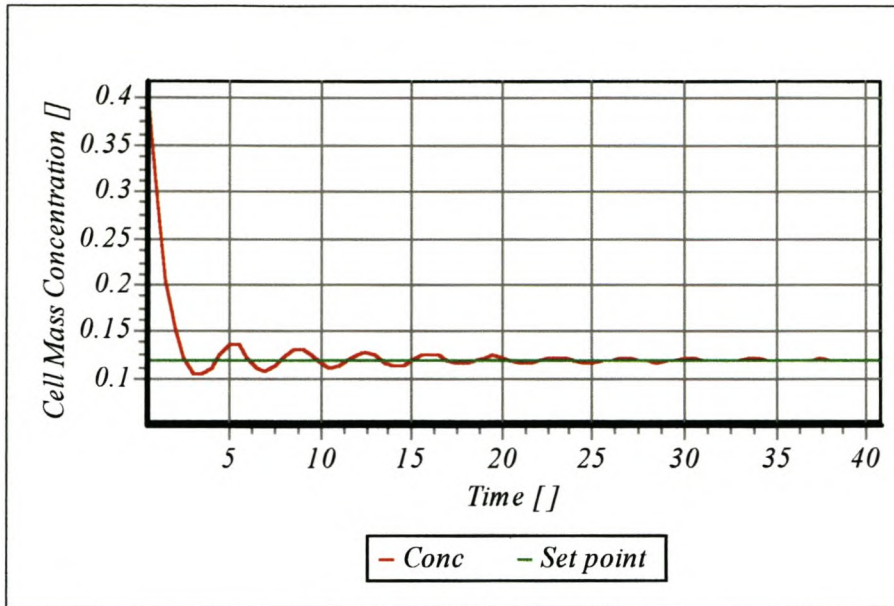


Figure 5-19 - Open loop response for the cell mass concentration with $w = 0.75$ []. Initial condition $C_1 = 0.65$ and $C_2 = 0.98$. A stable steady state point is reached $C_1 = 0.1207$ [] and $C_2 = 0.88$ [].

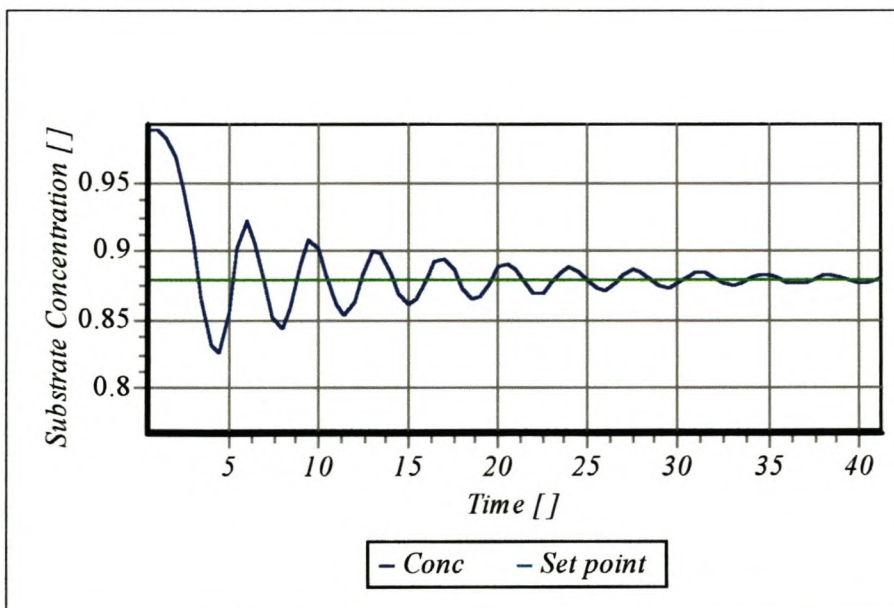


Figure 5-20 - Open loop response for the substrate conversion with $w = 0.75$ []. Initial condition $C_1 = 0.65$ [] and $C_2 = 0.98$ []. A stable steady state point is reached $C_1 = 0.1207$ [] and $C_2 = 0.88$ [].

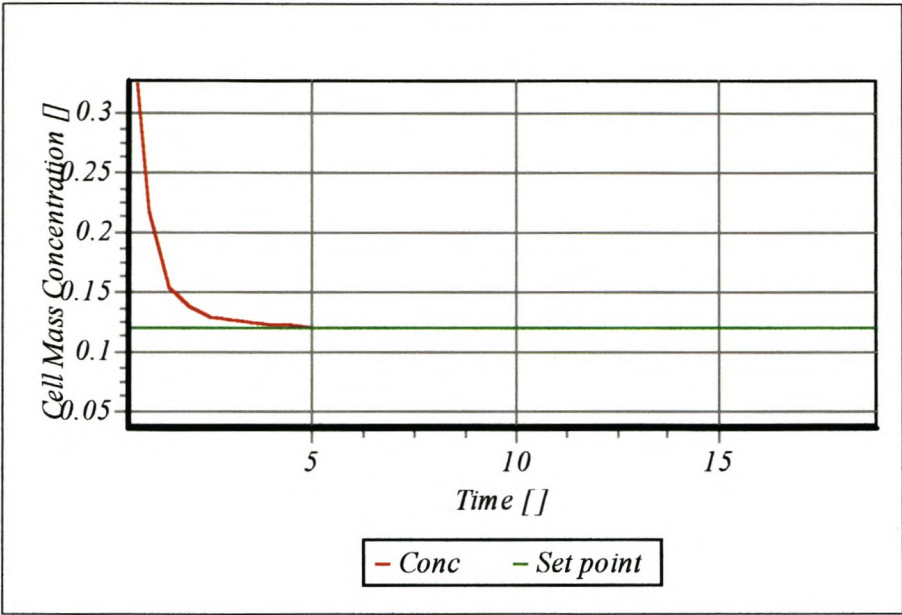


Figure 5-21 - Closed loop response for cell mass concentration to setpoint ($C_1 = 0.1207$ [], $C_2 = 0.88$ []) from initial condition $C_1 = 0.65$ [] and $C_2 = 0.98$ [].

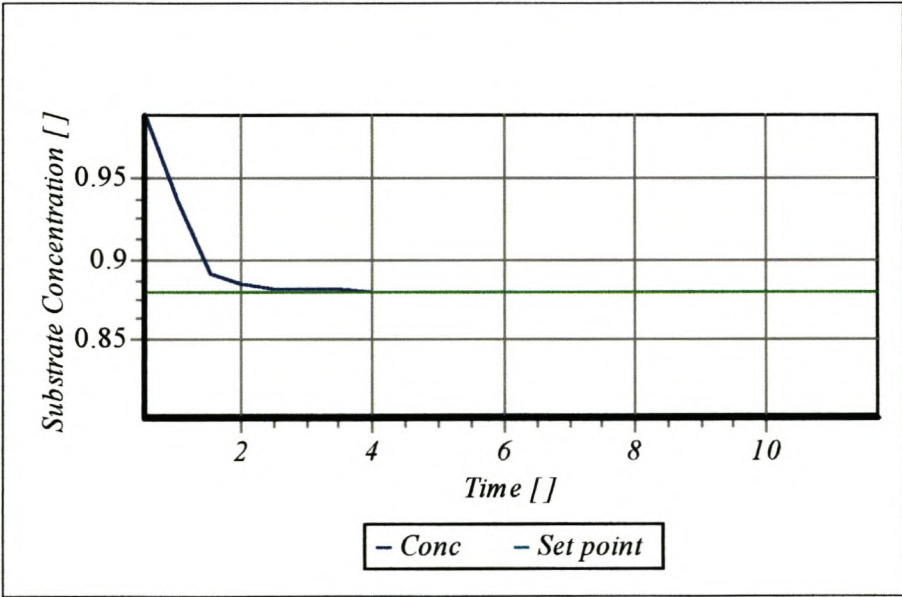


Figure 5-22 - Closed loop response for substrate conversion to setpoint ($C_1 = 0.1207$, $C_2 = 0.88$) from initial condition $C_1 = 0.65$ and $C_2 = 0.98$.

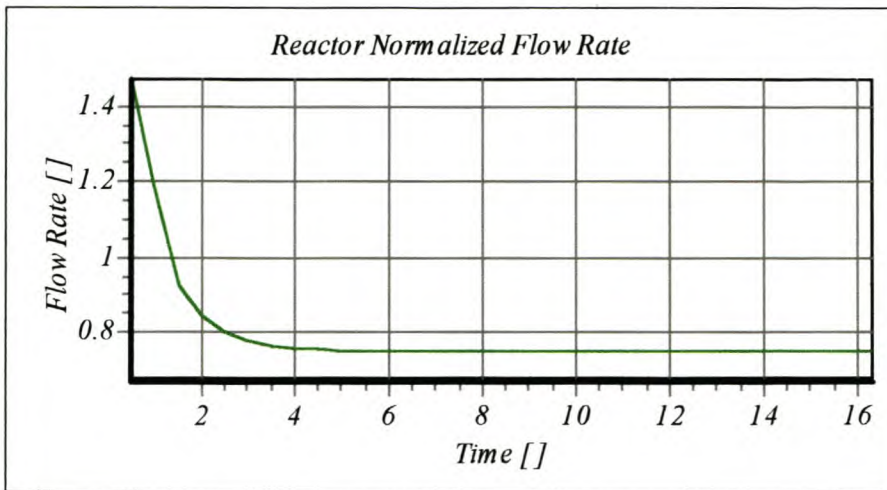


Figure 5-23 - Closed loop control action to setpoint ($C_1 = 0.1207$, $C_2 = 0.88$) from initial condition $C_1 = 0.65$ and $C_2 = 0.98$.

5.2.10.4 Servo Response Characteristics

A further desired control objective for the *neurocontroller* is to reach any desired set point without *state*. The *neurocontroller's* ability to reach untrained set points, demonstrates its ability to generalise to states not frequently (or ever) encountered during learning. The below figures 5-24 to figure 5-26 demonstrates the *neurocontroller's* ability to establish a set point change from a learned, stable steady state to an unlearned, unstable steady state point. This unlearned unstable steady state corresponds to the set point change investigated by Biegler & Seider (1989) (section 5.2.2). This is accomplished with a satisfactory transient response and negligible *state* of 0.6% in the cell mass concentration. The *neurocontroller* is thus able to accurately predict the steady state gain of the process at the unlearned desired set point. The neural *network* is thus able not only to deal effectively with complex dynamics, but also to maintain good servo characteristics by accurately predicting the steady state process gain.

Neural *networks* are particularly effective *generalisation* tools, particularly when interpolating to regions unseen during training or learning. As the desired set point is located between $SP_1(C_1 = 0.1207, C_2 = 0.88)$ and $SP_2(C_1 = 0.2107, C_2 = 0.7226)$ on the bifurcation analysis (figure 5-3), this accurate prediction of steady state gain results from effective interpolation by the *neurocontroller*.

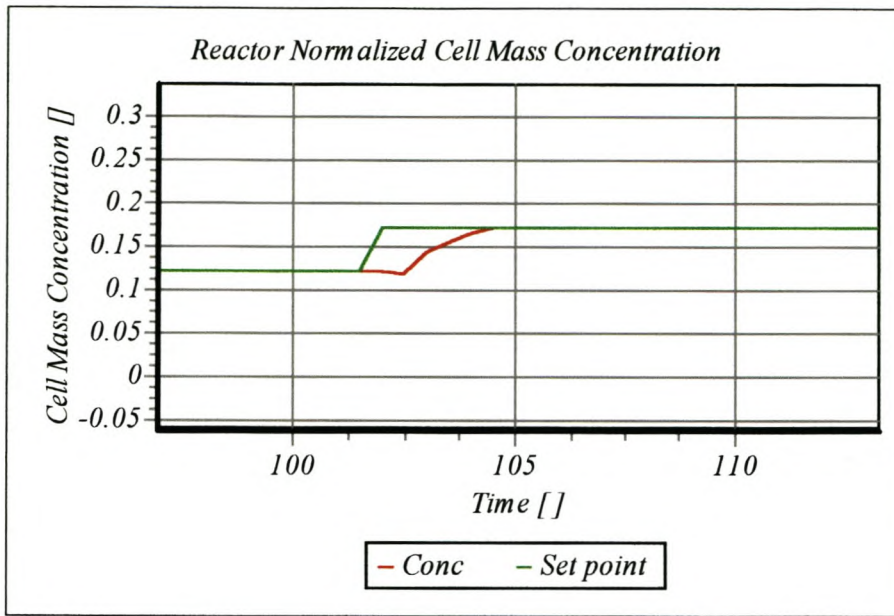


Figure 5-24 - Set point change cell mass concentration response from stable ($C_1 = 0.1207$, $C_2 = 0.88$) to untrained unstable steady state ($C_1 = 0.1707$, $C_2 = 0.8032$). Cell mass concentration steady state offset $\cong 0.001$ [].

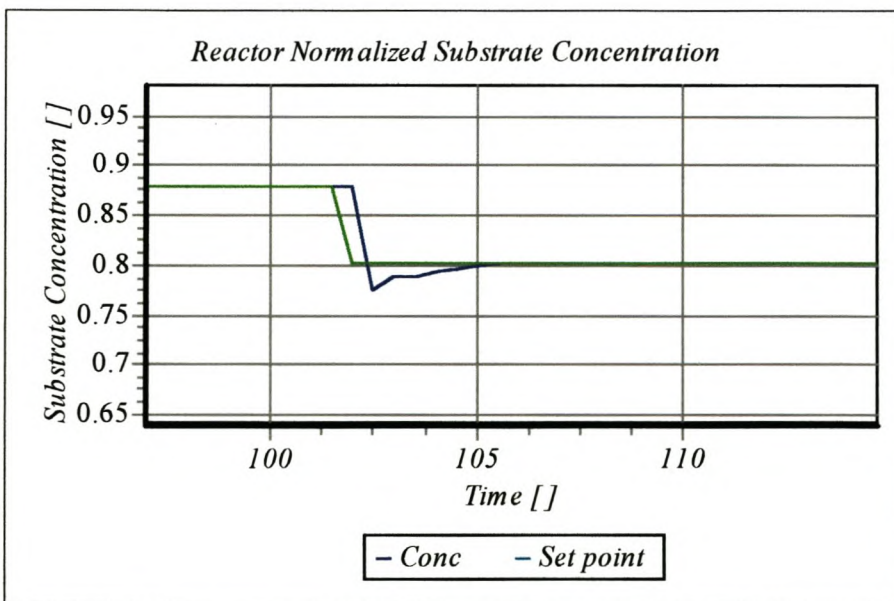


Figure 5-25 - Set point change substrate conversion response from stable ($C_1 = 0.1207$, $C_2 = 0.88$) to untrained unstable steady state ($C_1 = 0.1707$, $C_2 = 0.8032$). Substrate conversion steady state offset $\cong 0.002$ [].

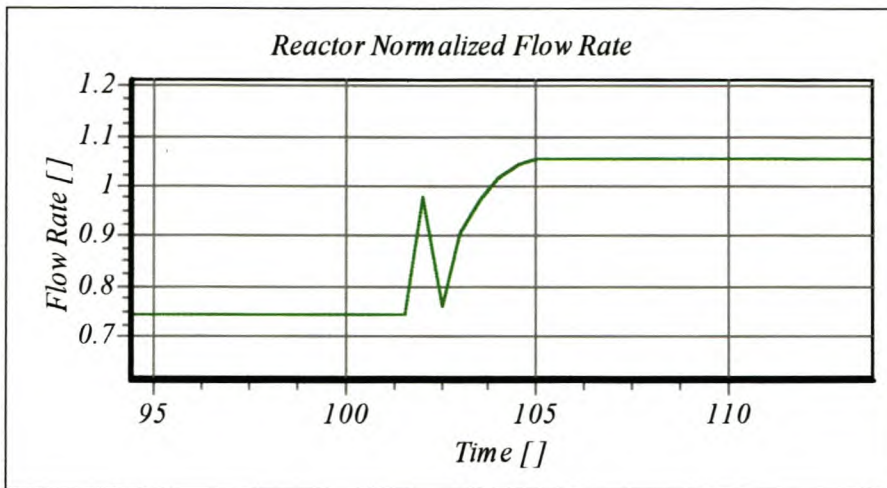


Figure 5-26 - Set point change control action from stable ($C_1 = 0.1207$, $C_2 = 0.88$) to untrained unstable steady state ($C_1 = 0.1707$, $C_2 = 0.8032$).

Harris et al. (1998) demonstrated the use of functional expansion models over a range of set point changes in the open loop stable steady state region (section 5.2.5). To further demonstrate the *generalisation* ability of the *neurocontroller* to unlearned set points, the set point changes described by Harris et al. (1998) are presented in figure 5-27 to figure 5-29. All three set point changes represent set points outside the learned experience of the *neurocontroller*.

The nominal set point $SP(C_1 = 0.0965, C_2 = 0.912)$ is reached with minimal offset, as well as the positive set point change from the nominal set point. Harris et al. (1998) reported slight oscillation for the FEx models for this positive set point change (figure 5-9). The controller based on a linear model suffered from sustained oscillation for this positive set point change (figure 5-9). From figure 5-27 and figure 5-28 no oscillatory behaviour is present, even from an initial condition that originates from the unstable region of operation.

For the negative set point change from the nominal steady state, the *neurocontroller* has offset amounting to a 2.5% offset in the cell mass concentration, and an offset of 0.54% in the substrate conversion set point. The FEx controller displayed an oscillatory response due to the approach to instability of the FEx model (figure 5-10). The linear controller remained robust for this negative set point change (figure 5-10).

The nominal steady state and the negative set point change investigated by Harris et al. (1998), both represent an extrapolation for the SANE developed *neurocontroller*. Although the *neurocontroller* remains able to maintain a non-oscillatory response in the closed loop dynamics, prediction of the steady state gain is less accurate than for interpolation. The minimal *state* indicates that the *neurocontroller* is able to remain robust, despite the need to extrapolate the steady state gain of the process.

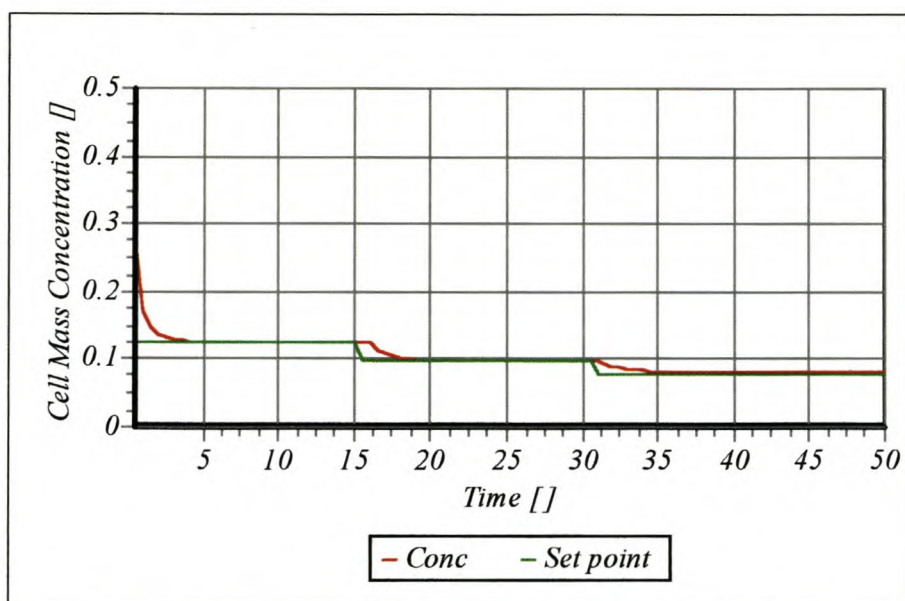


Figure 5-27 - Neurocontroller response to set point changes investigated by Harris et al. (1998). Steady state offset for final setpoint change ($C_1 = 0.0785$, $C_2 = 0.9345$) $\cong 0.002$ [].

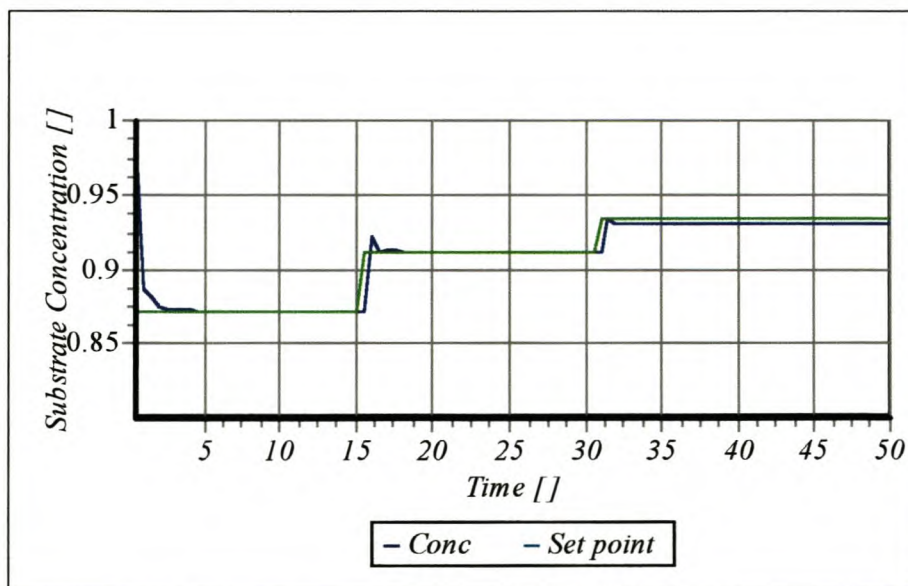


Figure 5-28 - Neurocontroller response to set point changes investigated by Harris et al. (1998). Steady state offset for final setpoint change ($C_1 = 0.0785$, $C_2 = 0.9345$) $\cong 0.005$ [] in substrate concentration.

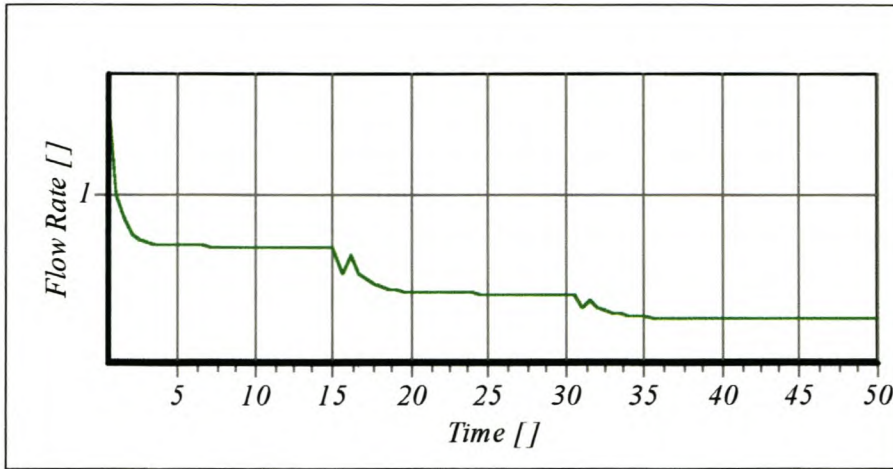


Figure 5-29 - Manipulated variable response for set point changes proposed by Harris et al. (1998).

5.2.10.5 Plant / Model Mismatch

As indicated by Henson & Seborg (1992) model parameters for fermenters may vary unpredictably over time. Brengel & Seider (1989) (section 5.5.2) considered an effective γ_{process} with fixed values of $\gamma_{\text{process}} = 0.47 - 0.49$, while γ_{model} was maintained constant at $\gamma_{\text{model}} = 0.48$. The resulting offset is illustrated (figure 5-5) and discussed in section 5.2.2. In order to evaluate the SANE *neurocontroller* in the presence of model mismatch in γ , an effective γ is calculated at each time step with a *gaussian* distribution (standard deviation of 0.01 [] in γ). The effective value of γ is illustrated in figure 5-30. Significantly in this analysis, is that the *neurocontroller* learned to control the fermentation process with a constant value for $\gamma = 0.48$. Any variation in γ thus represents a change in the *state* space not encountered during learning.

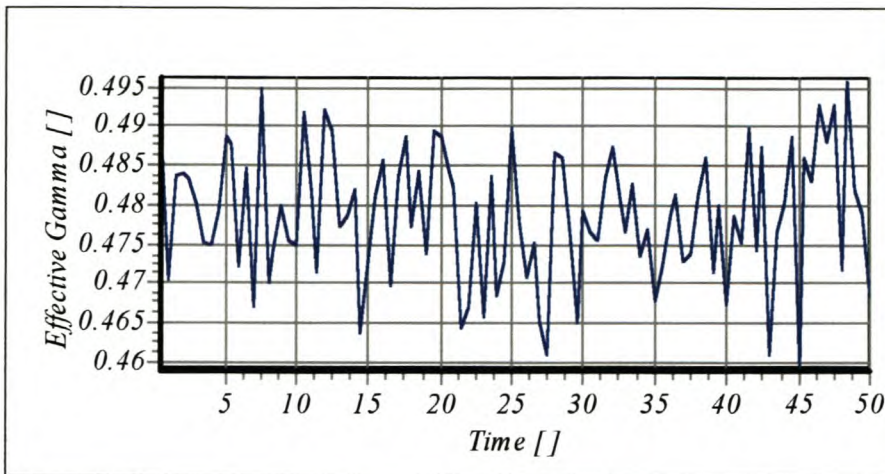


Figure 5-30 - Model $\gamma = 0.48$ for controller training. Effective γ assumed to be variable and dependent on process variables. Effective γ for each time step selected gaussian distribution around $\gamma = 0.48$ with STD = 0.01.

As illustrated in figure 5-31 to figure 5-33 the *neurocontroller* response rejects a variable γ effectively. This is in sharp contrast to the results reported by Brengel & Seider (1989), which indicate a 20% *state* (figure 5-5) without the incorporation of *model mismatch compensation*. Brengel & Seider (1989) utilised *model mismatch compensation* to eliminate the *state* as illustrated in figure 5-6. The *neurocontroller* is able to generalise and remain robust in the presence of large model parameter uncertainties in γ .

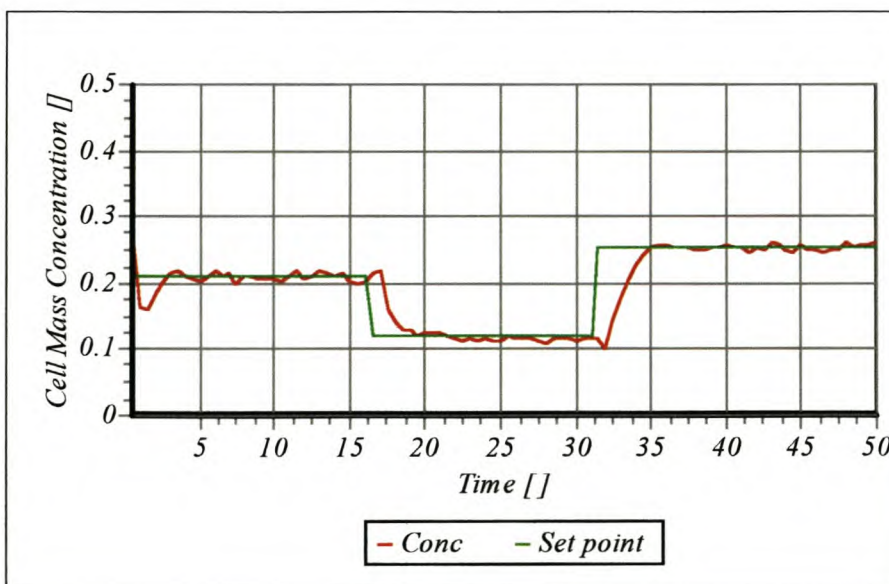


Figure 5-31 - Transient response for cell mass concentration with process / model mismatch in γ . At each time step the effective γ is assumed to be within a gaussian distribution from the mean $\gamma = 0.48$, with a STD of 0.01. This corresponds to > 2% error in the model parameter as described by Brengel & Seider (1989).

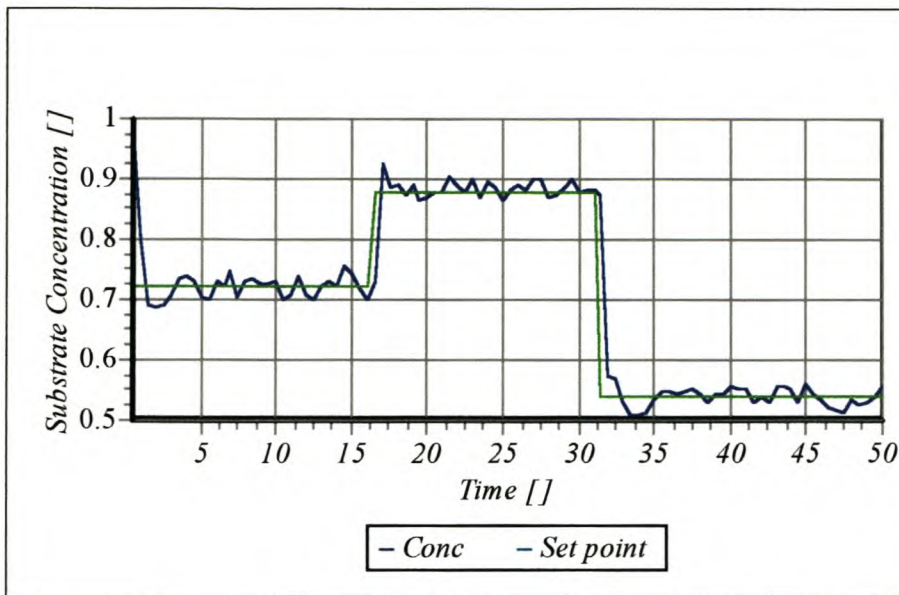


Figure 5-32 - Transient response for substrate concentration with process / model mismatch in γ . At each time step the effective γ is assume to be within a gaussian distribution from the mean $\gamma = 0.48$, with a STD of 0.01. This corresponds to > 2% error in the model parameter as described by Brengel & Seider (1989).

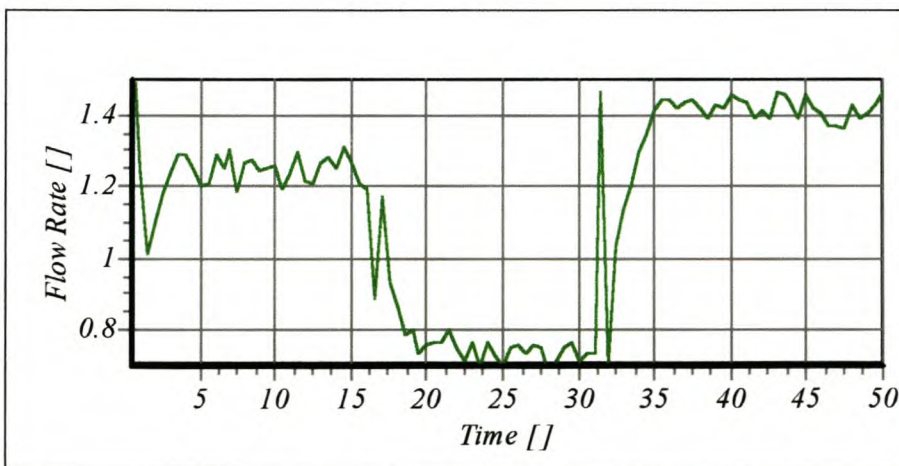


Figure 5-33 - Manipulated variable response with process / model mismatch in γ .

For 20% effective deviations in β at each timestep as illustrated in figure 5-34, the *neurocontroller* response is illustrated in figure 5-35 to figure 5-37. For 20% variations β , Brengel & Seider (1989) reported similar magnitudes of *state* as for 2% uncertainty in γ . Also, as discussed in section 5.2.2 *model mismatch compensation* cannot be used in model predictive control to eliminate the offset for variations in β . Adding an additional feed stream, with a different substrate concentration, would be required to allow for the use of *model mismatch compensation*. The SANE *neurocontroller* is, however, able to compensate for a varying β as illustrated in figure

5-35 and figure 6-36. No addition capital expenditure to add an additional feed stream is required to allow for varying β , as would be the case using model predictive control.

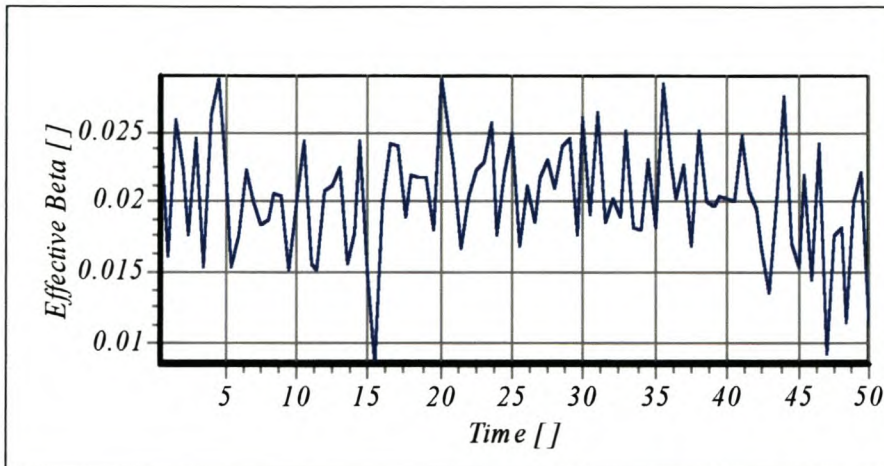


Figure 5-34 - Model $\beta = 0.02$ for controller training. Effective β assumed to be variable and dependent on process variables. Effective β for each time step selected with gaussian distribution around $\beta = 0.02$ with STD = 0.004.

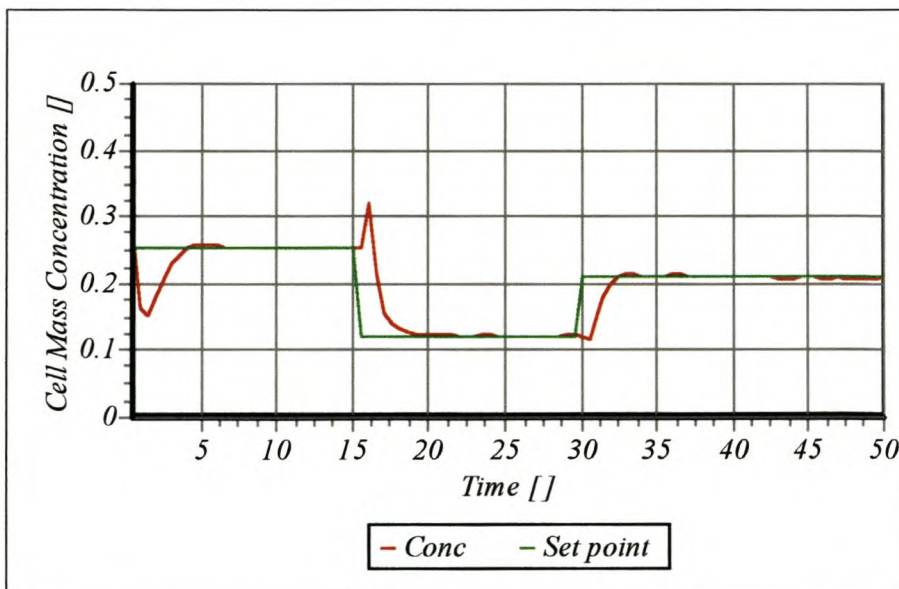


Figure 5-35 - Transient response for cell mass concentration with process / model mismatch in β . At each time step the effective β is assume to be within a gaussian distribution from the mean $\beta = 0.02$, with a STD of 0.004. This corresponds to > 20% error in the model parameter as described by Brengel & Seider (1989).

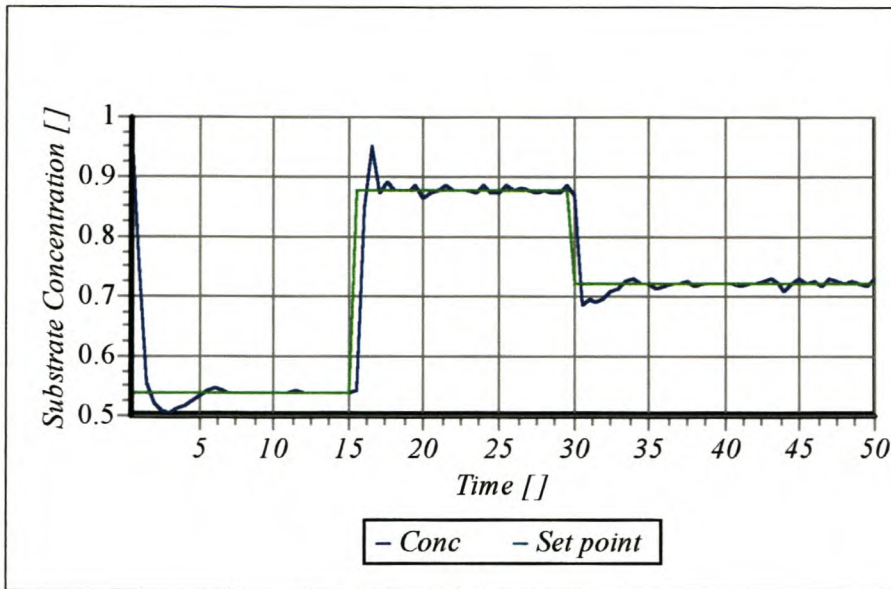


Figure 5-36 - Transient response for substrate concentration with process / model mismatch in β . At each time step the effective β is assume to be within a gaussian distribution from the mean $\beta = 0.02$, with a STD of 0.004. This corresponds to > 20% error in the model parameter as described by Brengel & Seider (1989).

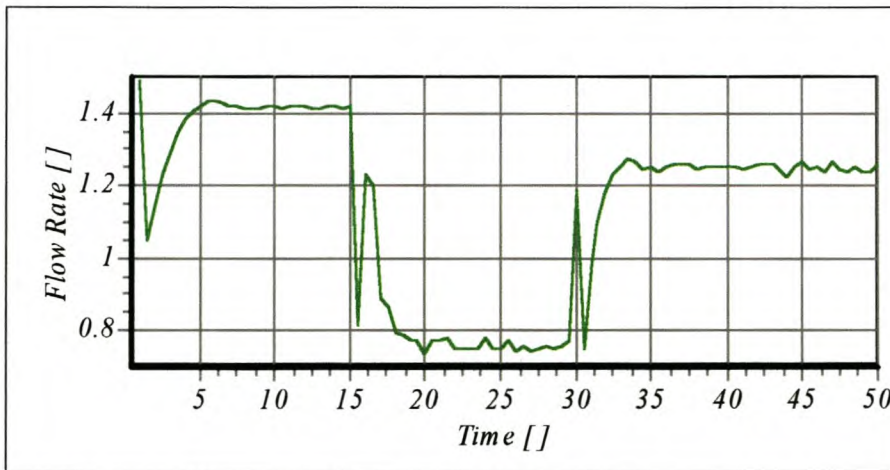


Figure 5-37 - Manipulated variable control action response with process / model mismatch in β .

Model predictive control is explicitly dependent on the process model in the control structure (section 4.1.5). Model parameter uncertainty may thus severely degrade the controller's performance. To compensate for this sensitivity, *process/model mismatch compensation* is employed as an additional external function in the optimisation algorithm. The *neurocontroller* is, however, able to remain robust in the presence of *process/model mismatch* by its implicit dependence on the process model. The process model is not required in the control loop as for model predictive control. It is

postulated that the *neurocontroller* learns inherent plant characteristics that reflect the dynamic model structure of the process, reducing the dependency of controller performance on explicitly defined model parameters. The physical structure of the model may be of greater relevance during *neurocontroller* behaviour learning than the actual model parameters reflected in the model structure. Model parameter uncertainty thus has less impact on the *neurocontroller's* performance, due to this implicit *generalisation*.

5.2.10.6 Unmeasured Disturbance Rejection

The ability of a controller to reject unmeasured disturbances is paramount to its effective implementation. As the continuous feed to the bioreactor may be variable due to upstream disturbances or a variable feedstock, a variable feed with standard deviation of 0.05 is introduced to the bioreactor as illustrated in figure 5-38. This is an occurrence not encountered during learning. This feed disturbance is effectively rejected in the cell mass concentration (figure 5-39), with poorer set point tracking in the substrate conversion (concentration). The sampling interval has a significant role in disturbance rejection, as the controller is only able to change its control action every 0.5 time units. In this case a new feed concentration disturbance is introduced every 0.5 time units, which does not allow for transients to decay to the set point. Should the sampling period be smaller than the feed disturbance interval, the *neurocontroller* will be able to track the set point more effectively. A worse case scenario is thus illustrated in the below figure 5-39 to figure 5-41, with frequent and large substrate feed concentrations disturbances.

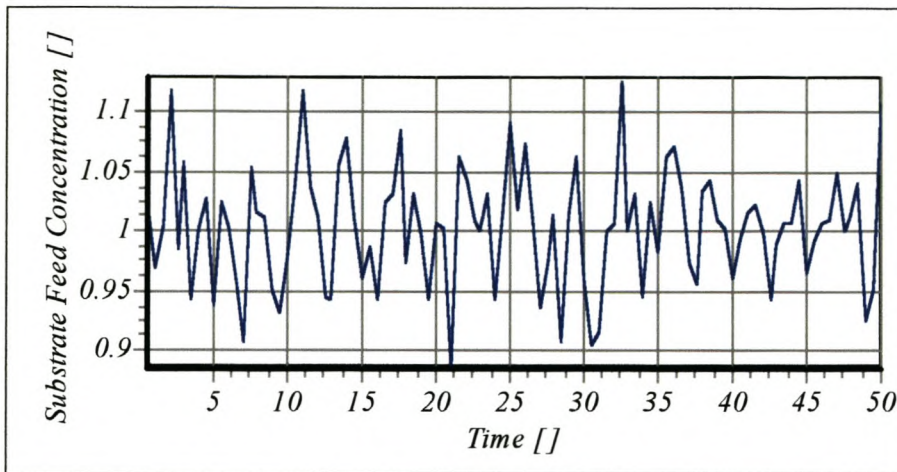


Figure 5-38 - Assumed variable substrate feed concentration with gaussian distribution around 1, with STD = 0.05.

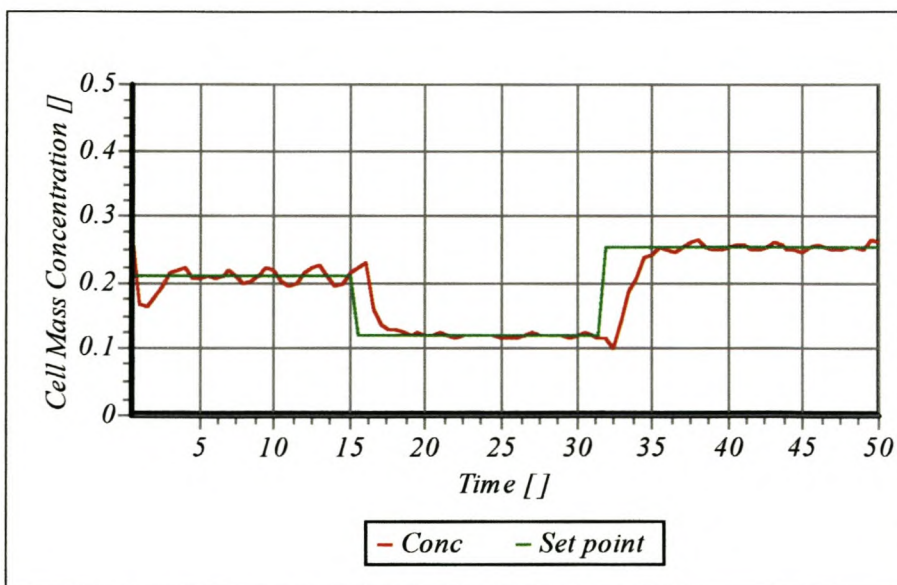


Figure 5-39 - Transient response for cell mass concentration with variable substrate feed concentration. Substrate feed assumed to vary in gaussian band around 1, with a STD = 0.05.

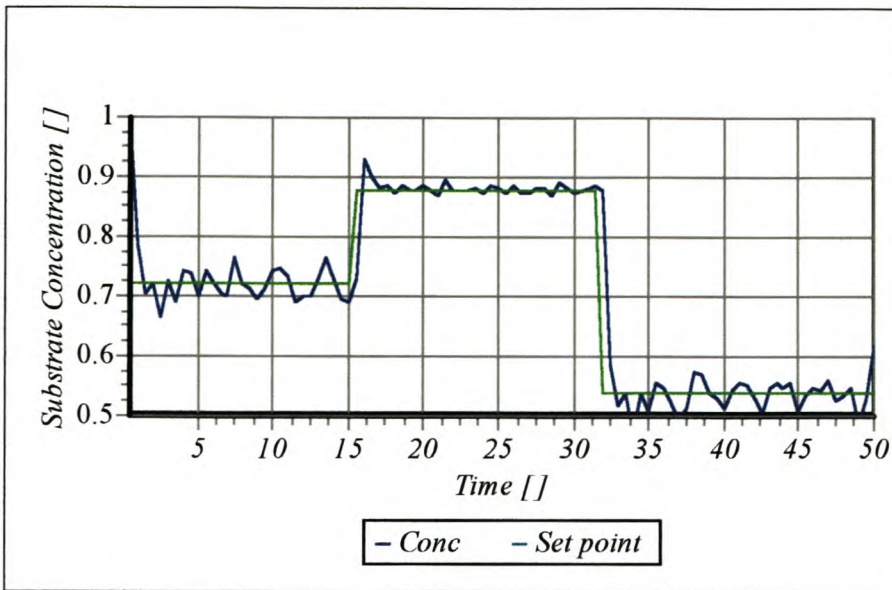


Figure 5-40 - Transient response for substrate concentration with variable substrate feed concentration. Substrate feed assumed to vary in gaussian band around 1, with a STD = 0.05.

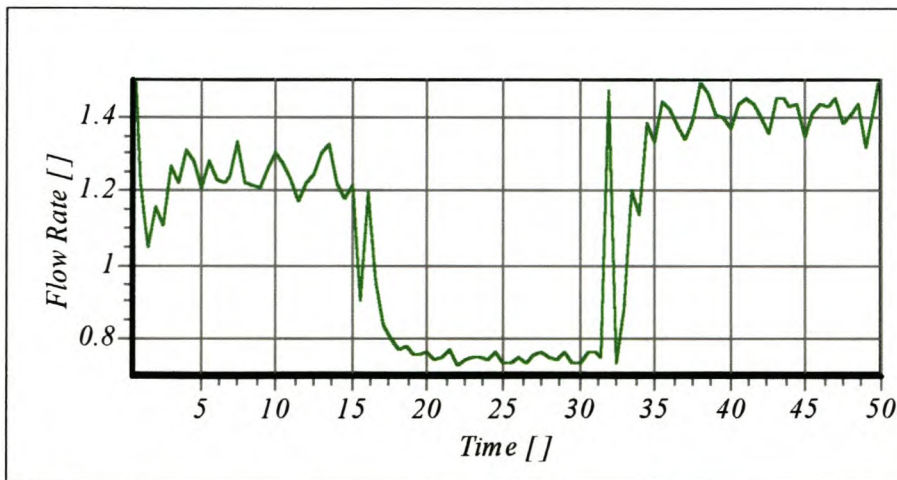


Figure 5-41 - Manipulated variable response to variable substrate feed concentration.

5.2.10.7 Controller performance with Sensor Noise

As indicated by Henson & Seborg (1992) biosensors are typically unreliable. Should reasonably accurate biosensors be unattainable, the necessary *process variables* are often inferred (i.e., from oxygen consumption). The ability of the *neurocontroller* to remain robust in the presence of measurement noise is thus paramount. Erroneous sensor readings may cause the process to become unstable should the model used in the control structure only be valid over a limited range of the *state* space (see section 5.2.5).

The transient response to a measurement noise with a standard deviation (*gaussian* distribution) of 0.02 [] from the actual cell mass concentration, is illustrated in figure 5-42 to 5-45. Figure 5-42 shows the actual and measured cell mass concentration. Even though the process may be accurately approximated by a linearised model in a limited operating range, measurement noise may indicate a process condition outside this operating range. This may resulting in a control action that is inappropriate. This may cause the process to drift from the applicable linearisation range, resulting in instability of the process. The *neurocontroller* with its wide range of robust performance may be expected to maintain effective performance, despite large measurement errors.

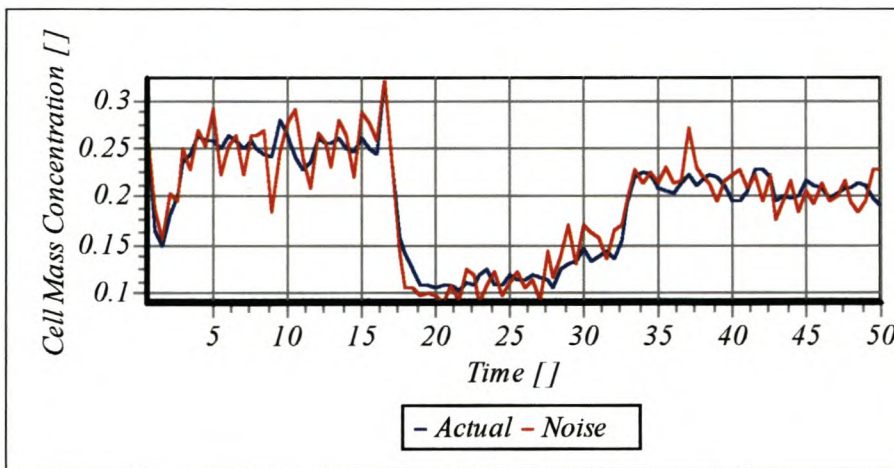


Figure 5-42 - Measured cell mass concentration (with noise) and actual cell mass concentration.

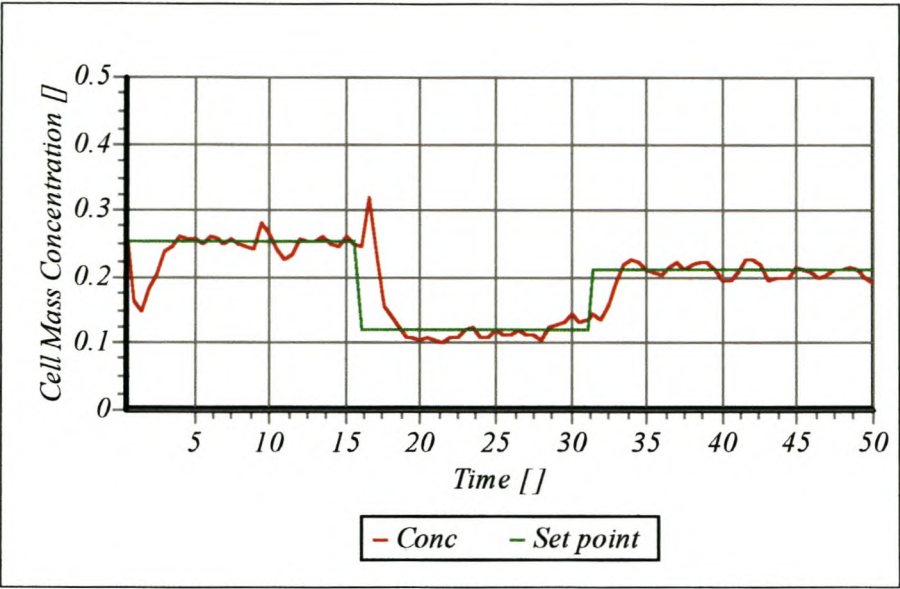


Figure 5-43 - Assumed sensor or inference noise for cell mass concentration. Gaussian noise assumed around actual cell mass concentration, with STD = 0.02. Effect on the cell mass concentration.

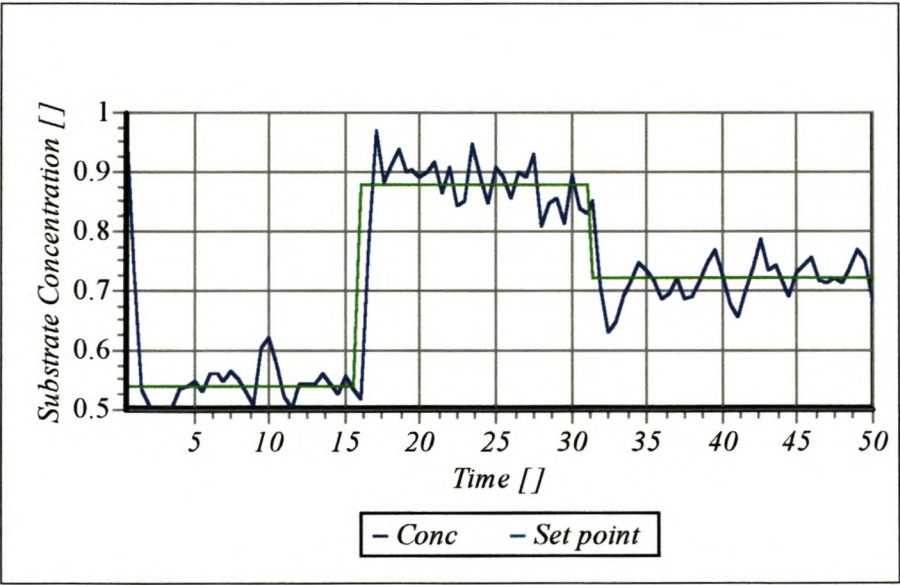


Figure 5-44 - Assumed sensor or inference noise for cell mass concentration. Gaussian noise assumed around actual cell mass concentration, with STD = 0.02. Effect on the substrate concentration.

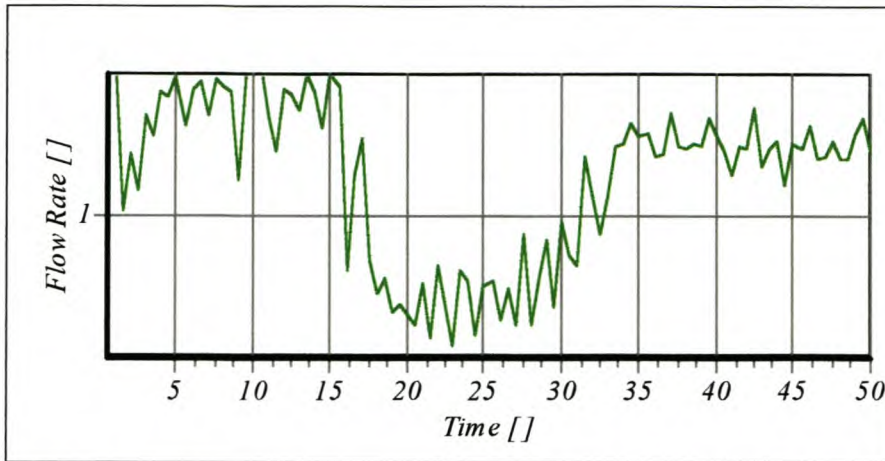


Figure 5-45 - Manipulated variable response to sensor or inference noise on cell mass concentration measurement.

5.2.11 Process Design and Control Optimisation Integration

Bregel et al. (1992) optimised the flowsheet for the Agrawal fermentation reactor, in order to maximise the venture capital of the process. From a operations point of view a conservative design may consider operation in the stable region prudent. The stable steady state that yields the highest cell mass is at the Hopf bifurcation point. Bregel et al. (1992), however, determined that the economic optimum is located in the unstable steady state region, with this economic optimum having a venture profit that exceeds the venture profit at the Hopf bifurcation analysis by 31.6%. As indicated by Seider et al. (1990) the coordination of process design, economic *evaluation*, operation and control optimisation is paramount. The flexibility of SANE in attaining this coordination ideal was investigated.

For effective design and control optimisation integration, it would generally be required to first utilise nonlinear analysis (nonlinear programming) techniques to determine the operating region of maximum economic benefit. Thereafter, an advanced controller (i.e., model predictive controller) may be developed that is able to perform optimally in this identified region of the *state* space (Seider & Bregel, 1990).

Utilising SANE for design and control optimisation integration analysis, no nonlinear analysis technique is required to find the operating region of maximum productivity,

before developing the controller. SANE is able to search the *state* space and find the maximum cell mass productivity that is obtainable, and simultaneously develop a optimal controller that is robust at this maximum cell mass productivity.

To find the region of maximum economic benefit, the set point criteria needs to be removed as inputs to each *neurocontroller*. SANE is thus required, as an additional task, to also find the set point that corresponds to maximum economic benefit. For the bioreactor, each *neurocontroller* thus consisted of two *state* inputs, 8 hidden nodes and a single *manipulated variable* output node. In this analysis it was assumed that the set point representing maximum cell mass productivity, was also the operating region of maximum economic benefit. In the analysis by Brengel et al. (1992), the maximum venture profit operating point, also corresponded to the point of maximum cell mass productivity. Should this not be the case, SANE's *fitness* function may be modified to include economic profit and operating costs per unit of cell mass produced.

SANE was consequently presented with the objective (*fitness* function) of finding a cell mass concentration $C_1 = 0.5$ [] and developing a controller to control the process at this discovered set point. From the bifurcation analysis in figure 5-3 it is evident that such a cell mass concentration is not an attainable steady *state* for the bioreactor as presented in section 5.2.1. SANE will thus only be able to reduce the error in the *fitness* function to an attainable minimum (maximum cell mass concentration).

SANE consequently developed a controller that performed optimally at the operating set point of SP($C_1 = 0.2542$, $C_2 = 0.5392$). From the bifurcation analysis (figure 5-3) it is apparent that this set point represents the maximum cell mass concentration attainable in the unstable steady state region. The task of optimal process design and control is thus effectively incorporated into a single calculation step. No prior nonlinear analysis is required to find the region of maximum venture capital (as performed by Seider & Brengel (1991)), before developing an advanced controller for this region of operation.

5.3 CATALYTIC REACTOR SIMULATION

Section 5.3.1 details the process description and section 5.3.2 to section 5.3.4 presents an overview of control strategies previously considered for the a catalytic reactor. Section 5.3.5 describes the control strategy developed using SANE.

5.3.1 Catalytic Reactor Process Description

The catalytic reaction $A + B \rightarrow P$ considered in this simulation study, is performed with A in excess. The rate of consumption of B, as described by Matsuura & Kato (1967) is -

$$r_b = \frac{k_1 \cdot C_b}{(1 + k_2 \cdot C_b)^2} \quad (5-8)$$

The general reaction rate equation described in equation 5-8, may be applied to the hydrogenation of ethylene in a continuous slurry reactor, should the hydrogen be in excess and the hydrogen molecule (not the dissociated hydrogen atoms) absorb onto the catalyst surface -

$$r_{C_2H_4} = -\frac{dC_{C_2H_4}}{dt} = \frac{k_1 \cdot C_{C_2H_4}}{(1 + k_2 \cdot C_{C_2H_4})^2} \quad (5-9)$$

where k_1 is the reaction rate constant and k_2 is the adsorption equilibrium constant. With hydrogen in excess the fluid volume is assumed not to be greatly influenced by the reaction. Also, the above reaction rate equation is only valid when the reactant concentration in the bulk fluid dominates the catalyst pores (Matsuura & Kato, 1967).

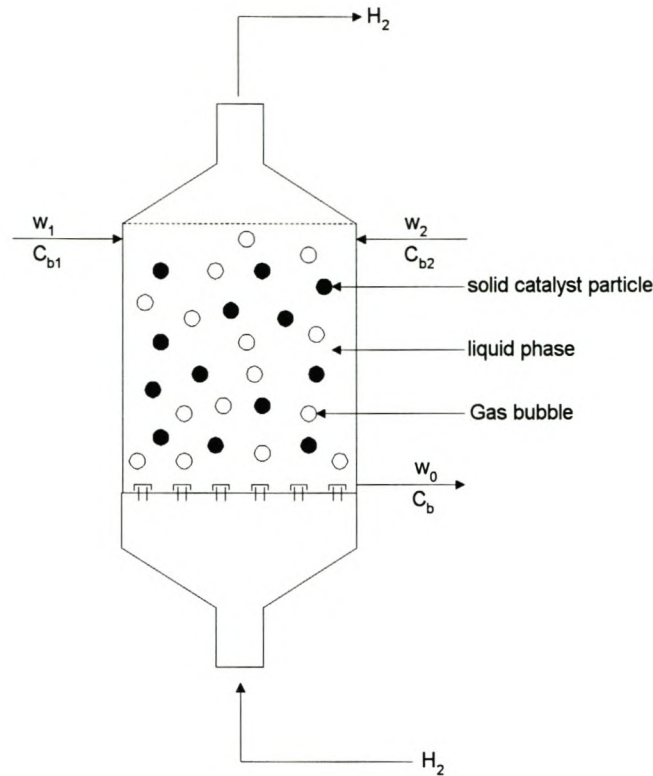


Figure 5-46 - Continuous Slurry Reactor with concentrated and dilute feed streams as manipulated variables.

The reaction is exothermic, but an excess rate of cooling is assumed to allow for isothermal operation of the reactor. Figure 5-46 illustrates a configuration proposed to control the product concentration, C_b , and the liquid level, h , by adjusting the flow rate, w_1 , of the concentrated feed (C_{b1}) and the flow rate, w_2 , of the dilute feed (C_{b2}). This configuration has the following dynamic model as proposed by Li & Biegler (1988), with the model parameters described in table 5-3.

$$\frac{dh}{dt} = w_1 + w_2 - 0.2 \cdot h^{0.5} \quad (5-10)$$

$$\frac{dC_b}{dt} = (C_{b1} - C_b) \cdot \frac{w_1}{h} + (C_{b2} - C_b) \cdot \frac{w_2}{h} - \frac{k_1 \cdot C_b}{(1 + k_2 \cdot C_b)^2} \quad (5-11)$$

Table 5-3 - Model parameters for the dynamic model of the isothermal slurry reactor.

<i>Description</i>	<i>Formulation</i>	<i>Unit</i>
Concentrated Feed (B)	$C_{b1} = 24.9$	$\left[\frac{\text{mol}}{\text{dm}^3} \right]$
Dilute Feed (B)	$C_{b2} = 0.1$	$\left[\frac{\text{mol}}{\text{dm}^3} \right]$
Reaction Rate Constant	$k_1 = 1$	$\left[\frac{\text{dm}^3}{(\text{mol} \cdot \text{min})} \right]$
Adsorption Equilibrium Constant	$k_2 = 1$	$\left[\frac{\text{dm}^3}{\text{mol}} \right]$
Steady State Concentration B	$C_{bs} = 2.7927$	$\left[\frac{\text{mol}}{\text{dm}^3} \right]$
Steady State Liquid Level	$h_s = 100$	$[\text{dm}]$

Assuming a constant volume reactor ($\frac{dh}{dt} = 0$), the mass balance equation has three different steady state solutions (α, ϕ, γ), of which α and γ are open loop stable and ϕ open loop unstable (concentration unstable). When the reactant concentration B drops slightly from $C_{B\alpha}$ to $C_{B\phi}$ (figure 5-47) for the nominal case, the mass balance equation is not satisfied. The supply of reactant is greater than the consumption in the reactor and the reactant concentration C_B increases, until the stable point α is reached. Similar reasoning follows at stable point γ . At the unstable point ϕ , should the reactant concentration decrease slightly below ϕ , it will continue to decrease until the stable point α is reached. Similarly, a slight increase in reactant concentration at unstable point ϕ , causes the reactant concentration to increase until the stable point γ is reached.

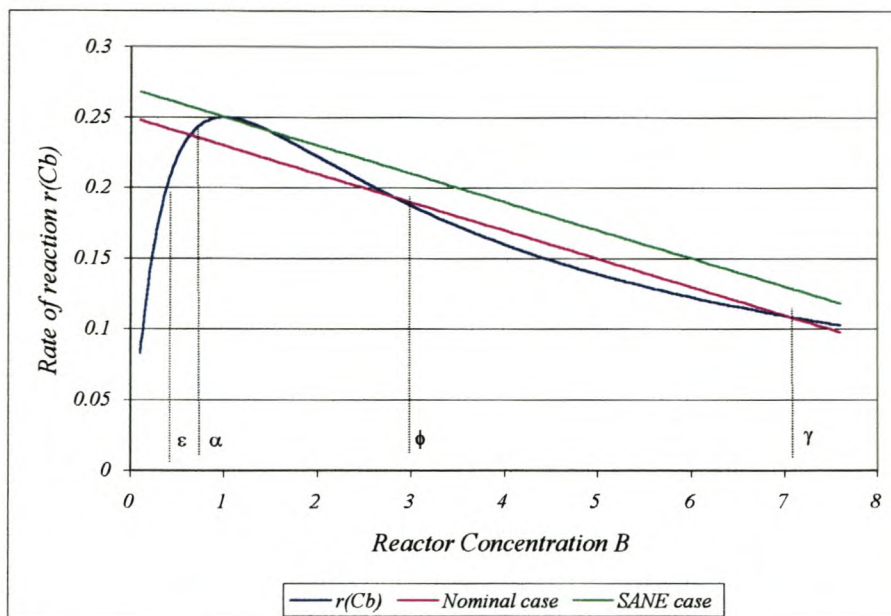


Figure 5-47 - Multi equilibrium points at steady state.

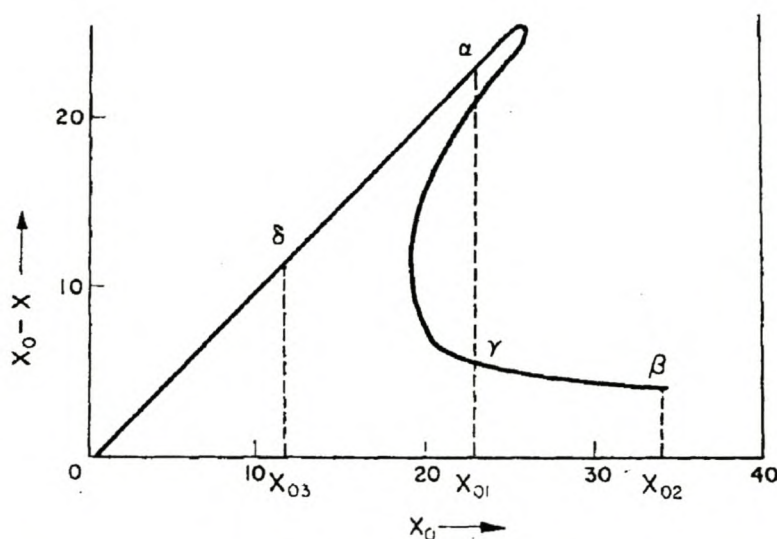


Figure 5-48 - Reactor space time yield as a function of the overall inlet reactant concentration (X_0) (Matsuura & Kato, 1967).

Consider the reactor being operated in the open loop (figure 5-48). The reactor is at first operated at the point α (X_{01}), where the space time yield is almost maximum and the reactor capacity is accordingly large. When the mixing rate of the fluid makes an abnormal change and X_0 suddenly reaches X_{02} (figure 5-48) the reactor space time yield becomes very low as it reaches β . Should X_{01} again be obtained, the point α can not be attained and the reactor operates at the point γ . It would appear as if the catalyst

has lost its activity, though the true phenomenon is a result of too much adsorption of the one reactant onto the catalyst surface, causing desorption of the other reactant. This results in poor overall conversion (Matsuura & Kato, 1967).

The catalyst surface is so-called "wetted" by the reactant. In order to recover the activity the catalyst surface must be "dried". X_0 must be lowered to X_{03} , from where it may be raised again to X_{01} , again attaining operating point α . Matsuura & Kato (1967) make two suggestions to overcome this difficulty: (1) the catalyst may be assumed deactivated and discarded; (2) robust control (for controlling mixing & feed rate disturbances) needs to be applied to ensure stable operation at the maximum space time yield.

Furthermore, robust control is complicated by the system gain sign change (illustrated in figure 5-47), as the operating point changes from one side of the unstable point ϕ to the other. As indicated by Economou & Morari (1986) systems of this type are not "integral controllable"; system stability may not be maintained over the entire operating region when controllers with integral action are employed. Without integral action *state* will result. An advanced control structure is thus required to maintain robust performance.

5.3.2 Single Step pseudo-Newton Model Predictive Control algorithm

For the nominal mass balance solution (graphical solution in figure 5-47) proposed by Li & Biegler (1988), three steady state attractors (two stable and one unstable) exist when $w_1 = 1 \left[\frac{dm}{s} \right]$ and $w_2 = 1 \left[\frac{dm}{s} \right]$. The controllability of their single step algorithm in reaching the unstable steady state, $h_s = 100 [dm]$ and $C_{bs} = 2.7927 \left[\frac{mol}{dm^3} \right]$ (operating point ϕ), from an initial condition (reactor start-up) at both sides of the unstable steady state ($h = 40 [dm]$ & $C_b = 0.1 \left[\frac{mol}{dm^3} \right]$; $h = 40 [dm]$ & $C_b = 7.0 \left[\frac{mol}{dm^3} \right]$) was investigated. The sampling time is chosen as $T = 1.0 [min]$. Various bounds on the flow rate manipulated variable were considered. The transient response for the

level and concentration of B are illustrated in figure 5-49, indicating the impact of various *manipulated variable* bound ranges on the response. Bounds on the inlet flow rates of 0 and 10 are relevant to this study.

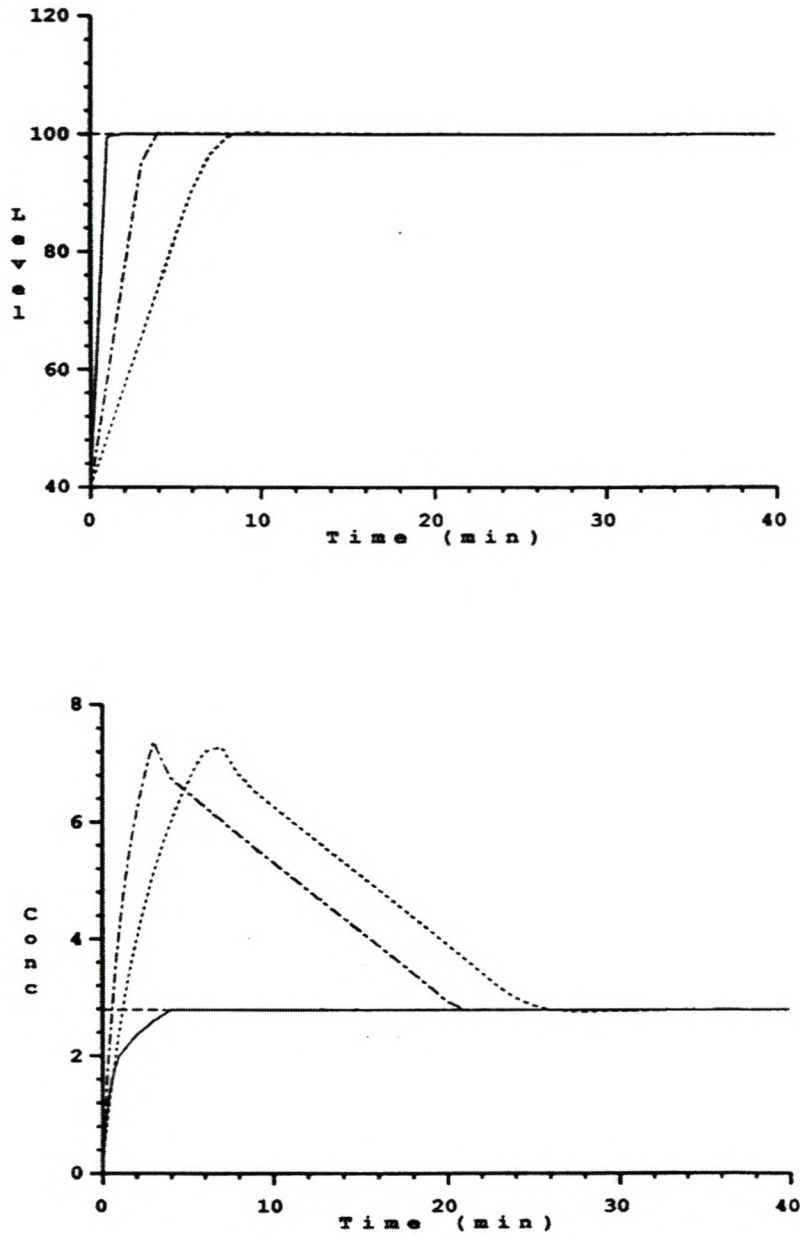


Figure 5-49 - Transient response from initial conditions of $C_b = 0.1$ and $h = 40$. Solid line, no upper bounds on u_1 and u_2 ; alternate dots and dashes, upper bound of 10 on u_1 and u_2 ; dotted line, upper bound at 5 on u_1 and u_2 (Li & Biegler, 1988).

5.3.3 Multi-Step Nonlinear Predictive Controller

Bregel & Seider (1989) also considered the servo response in reaching the unstable reactor steady state (ϕ) from the initial conditions as specified by Li & Biegler (1988) (section 5.3.2).

As seen in figure 5-50 Bregel & Seider (1989) reported a marked improvement with a prediction horizon of 4 steps, over the results obtained with Li & Biegler's (1988) single step algorithm. The settling time and overshoot is significantly reduced, when compared to figure 5-49. Figure 5-51 illustrates the impact a 20% model error in C_{b1} . Without model/process mismatch compensation significant offset results. The parameter estimation algorithm is able to eliminate the offset.

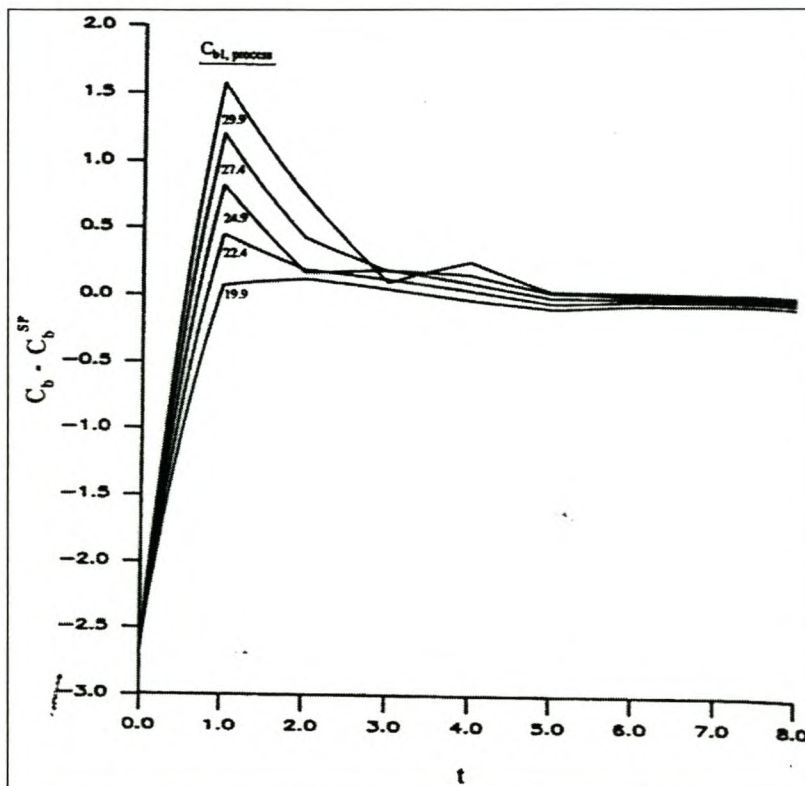


Figure 5-50 - Transient response to reactor start-up condition ($C_b = 0.1$, $h = 40$) (Bregel & Seider, 1989).

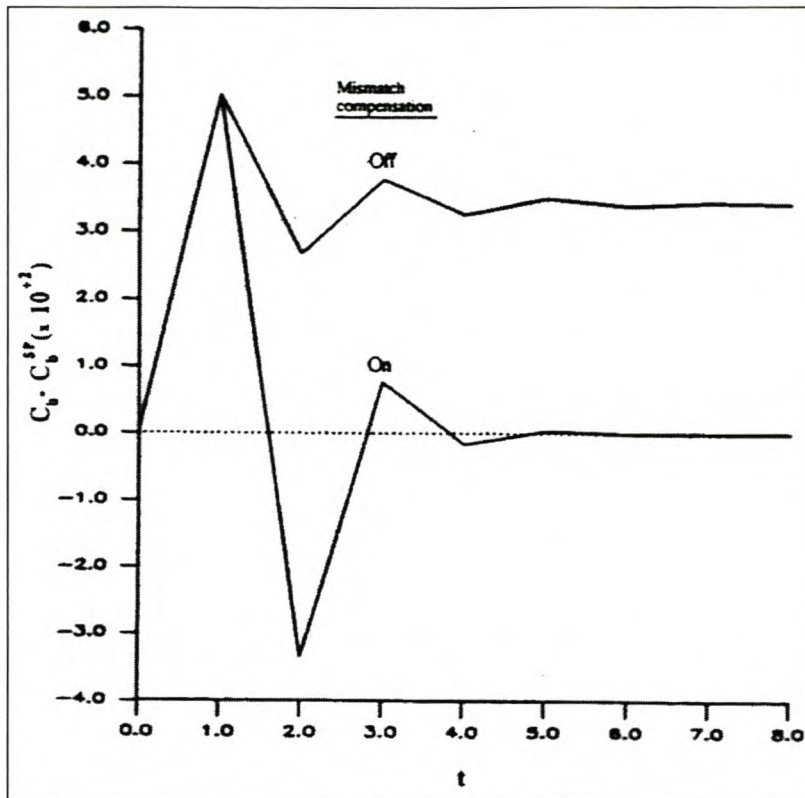


Figure 5-51 - Transient response to reactor start-up with process / model mismatch in C_{b1} of 20%, also showing the effect of mismatch compensation (Bregel & Seider, 1989).

5.3.4 Nonlinear Quadratic Dynamic Matrix Control with State Estimation

Gattu & Zafiriou (1992) investigated the servo performance of their Nonlinear Quadratic Dynamic Matrix Control in reaching the unstable steady state from the initial conditions proposed by Li & Biegler (1988). It is shown that the closed loop performance is significantly effected by the weight values selected in the diagonal weight vector (Γ) of the algorithm. Figure 5-52 & figure 5-53 show how the performance of the closed loop is effected based on the selected weight values. For a weight vector $\Gamma = \text{diag}[1,1]$ a transient response similar to that obtained by Li & Biegler (1988) (section 5.3.2) was obtained. For a weight vector $\Gamma = \text{diag}[1,20]$ a transient response similar to that obtained by Bregel & Seider (1989) (section 5.3.3) was obtained. A sampling period of $T=1.0$ [min] was utilised and the *manipulated variables* were bounded between 0 and 10.

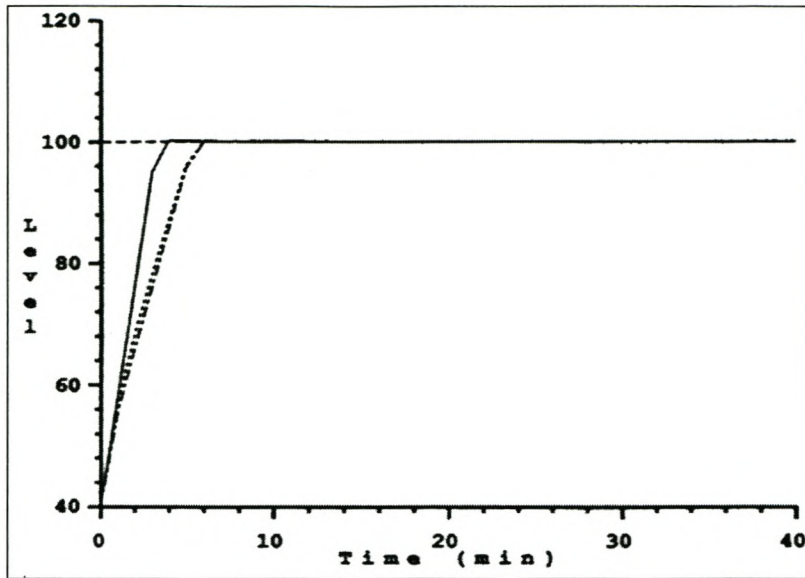


Figure 5-52 - Transient level response for reactor start-up response ($C_o = 0.1$; $h_o = 40$). Solid line, $\Gamma = \text{diag}[1,1]$; dotted line, $\Gamma = \text{diag}[1,10]$; alternate dots and dashes, $\Gamma = \text{diag}[1,20]$ (Gattu & Zafiriou, 1992).

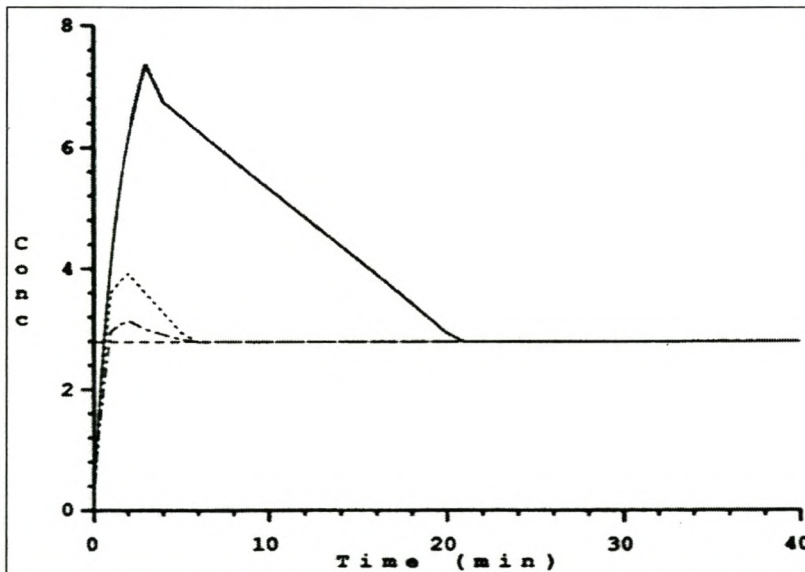


Figure 5-53 - Concentration response for reactor start-up response ($C_o = 0.1$; $h_o = 40$). Solid line, $\Gamma = \text{diag}[1,1]$; dotted line, $\Gamma = \text{diag}[1,10]$; alternate dots and dashes, $\Gamma = \text{diag}[1,20]$ (Gattu & Zafiriou, 1992).

5.3.5 Concluding remarks for control strategies considered in previous research

The catalytic reactor considered by Li & Biegler (1988) is a useful benchmark for controller *evaluation*, as the dynamic system includes multiple steady states (two stable and one unstable), and a process gain sign change around the unstable steady state. The model predictive control implementations, as for the bioreactor in section 5.2, prove sensitive to model parameter uncertainty. In this case, a 20% change in the unmeasured feed concentration resulted in the process drifting from the unstable set point to the a stable operating point. *Model mismatch compensation* was, however, able to eliminate the offset. Also, the significant dependence of the closed loop response due to *MPC* weighting vector tuning is evident, which implies the optimization of the weight vector parameters is paramount in any design.

5.3.6 Symbiotic, Adaptive Neuro Evolution (SANE)

The three cited literature studies for the control of the reactor (section 5.3.2 to 5.3.4), consider only the control around the unstable steady state ϕ . In order to demonstrate the performance of new control algorithms, this is a plausible control objective as effective control performance around this point implies an ability to deal with the process gain sign change at either side of ϕ . This operating point is not optimal from an economic viewpoint, as the reaction rate is not a maximum at this point. A slightly different control objective is proposed, with equal control difficulty, while maintaining the reactor at the operating point of maximum (economic) reaction rate.

As the integration of optimal process design and optimal controller development (section 5.2.10) is a significant benefit provided by the *SANE* algorithm, this development approach was adopted in the design of an optimal economic *neurocontroller* for the catalytic reactor. It was assumed that the most desired operation state for the reactor, is at the maximum attainable space time yield. The y-axis in figure 5-47 corresponds to the rate of reaction or consumption of reagent B. The curve for the reaction rate as a function of C_B has a maximum reaction rate between the stable steady state α and the unstable steady state ϕ solutions (nominal

case in figure 5-47). To attain this maximum point the straight line in the graphical solution of the reactant mass balance equations, needs to intersect this maximum point. The straight line intercepts the y-axis at $\frac{C_{B0}}{\tau}$. Maintaining the desired reactor level ($h = 100$) as a set point fixes the reactor space time at a constant value, also fixing the gradient of the straight line ($\frac{1}{\tau}$). As the proposed process configuration (figure 5-46) scheme allows for a concentrated and a dilute stream in the reactant B, the effective inlet C_{B0} may be considered variable, depending on the flow rates of the concentrated and dilute feed streams. In the graphical solution this implies that the straight line may be moved vertically with constant gradient, by adjusting the feed flow rates appropriately (figure 5-47).

In order to move the stable steady state α to the maximum reaction rate optima, the straight line in figure 5-47 needs to intercept the y-axis at a higher point (higher effective C_{B0}). With α at the maximum reaction rate, the unstable steady state ϕ moves closer to α , as the differential C_B between the two steady states is reduced. This is evident from an examination of the nominal and *SANE* graphical solutions in figure 5-47. The stable steady state γ moves further from ϕ , as the differential C_B between the two steady states is increased.

It is desirable to operate at α from an economic viewpoint (higher reaction rate). The closer proximity in the unstable steady state ϕ to the stable steady state α , implies that smaller process disturbances effecting C_B are required for the process to exhibit a process gain sign change. This complicates control and requires the use of advanced control techniques.

The *SANE* algorithm was utilised to simultaneously (as in section 5.2.10) find the operating region of maximum reaction rate in the state space and develop a *neurocontroller* for optimal control. A level set point of $h = 100$ [dm] was incorporated as a process constraint in the search for the optimum operating point. The additional performance criteria required *SANE* to search the state space for the point of maximum rate of reaction - no set point for C_B was provided. *SANE* was able

to find the operating point of maximum reaction rate without any prior nonlinear analysis of the state space, and develop a high performance controller that is robust over a wide range of the state space. The developed *neurocontroller*'s performance is illustrated in sections 5.3.6.1 to 5.3.6.3.

5.3.6.1 Learning optimal behaviour

SANE was presented with a limited range of initial conditions from which to learn to control the reactor. The initial concentration of B in the reactor was randomly selected with a *gaussian* distribution around a mean concentration $C_b = 1.713 \left[\frac{\text{mol}}{\text{dm}^3} \right]$, with a standard deviation of $0.3 \left[\frac{\text{mol}}{\text{dm}^3} \right]$. The liquid level in the reactor was randomly selected with a *gaussian* distribution around a mean of $h = 100 \text{ [dm]}$, with a standard deviation of 5 [dm] . For the best evolved *neurocontroller* the transient response from an initial condition in the learning region of the state space is illustrated in figure 5-54 to figure 5-57.

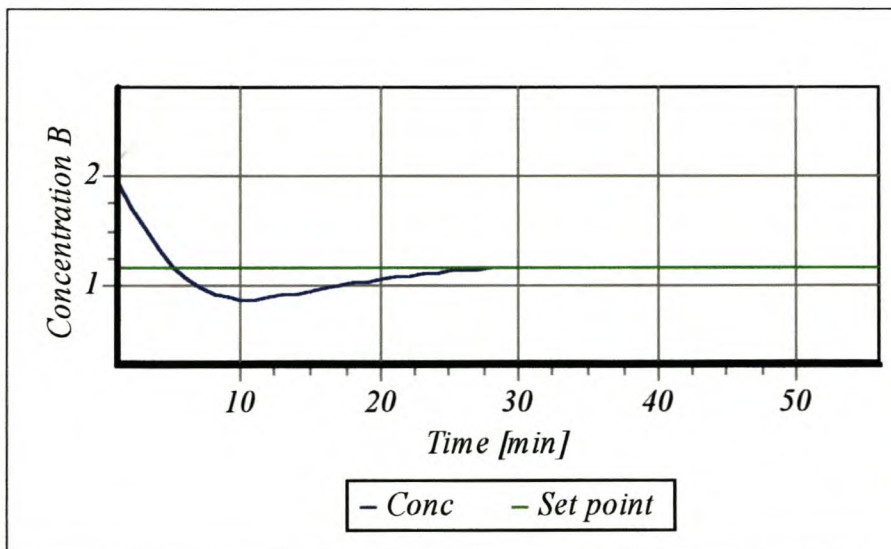


Figure 5-54 - Transient response for the concentration of reactant B, from initial reactor condition $C_b = 1.97$ and $h = 103.43$.

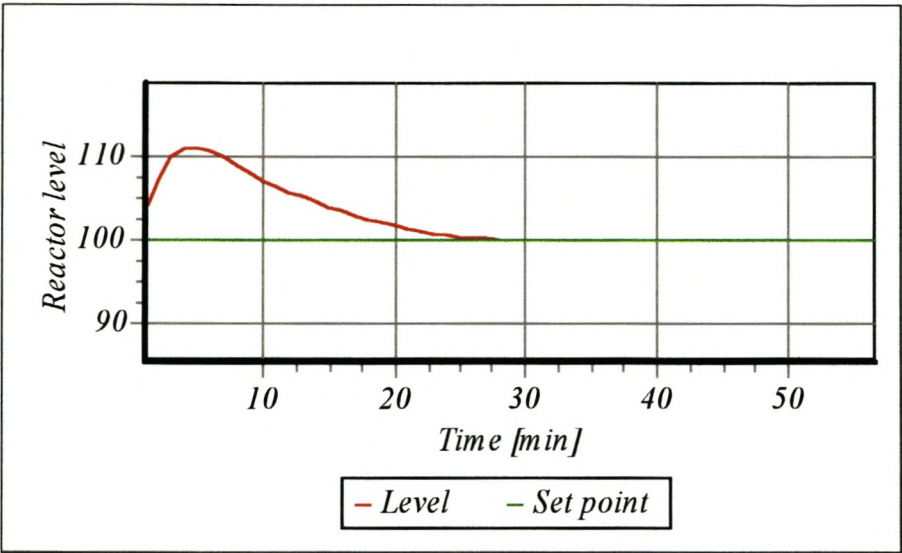


Figure 5-55 - Transient response for the reactor level, from initial reactor condition $C_b = 1.97$ and $h = 103.43$.

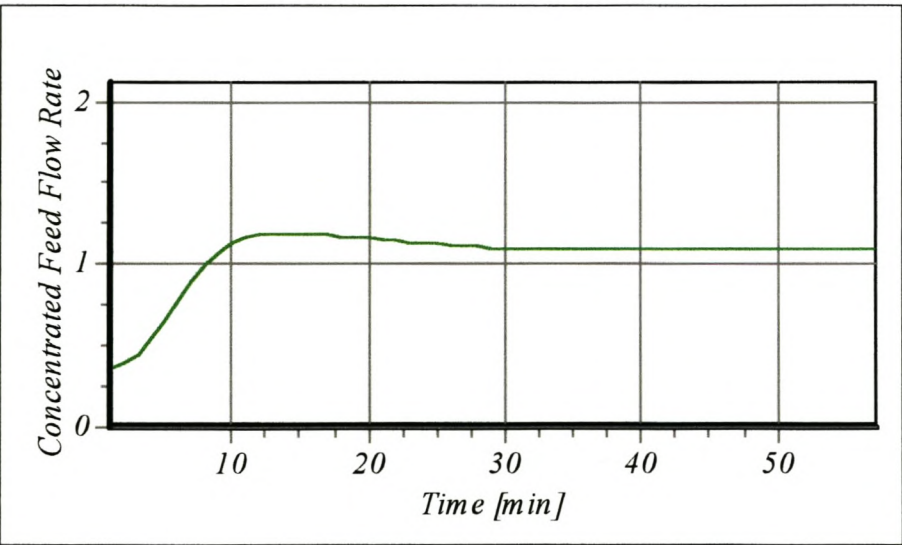


Figure 5-56 - Concentrated feed flow rate control action from initial condition $C_b = 1.97$ and $h = 100$ to set point.

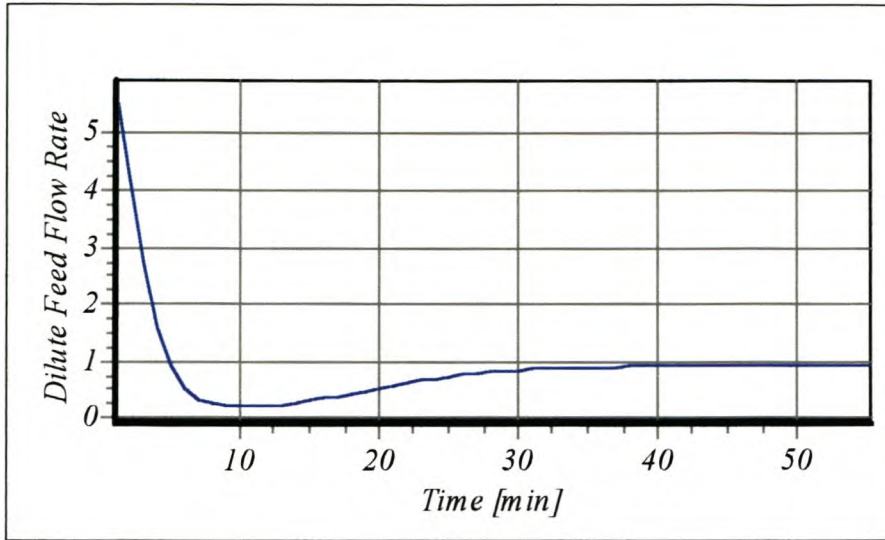


Figure 5-57 - Dilute feed flow rate control action from initial condition $C_b = 1.97$ and $h = 100$ to set point.

5.3.6.2 Servo Characteristics

As considered in sections 5.3.2, 5.3.3 and 5.3.4 the developed controller's servo performance was investigated from two initial conditions ($C_b = 0.1 \left[\frac{\text{mol}}{\text{dm}^3} \right]$, $h = 40 \text{ [dm]}$) and ($C_b = 7.0 \left[\frac{\text{mol}}{\text{dm}^3} \right]$, $h = 40 \text{ [dm]}$) corresponding to either side of the unstable steady state ϕ . These two initial conditions correspond to initial states that were not encountered during learning. The ability of the *neurocontroller* to generalise to unencountered initial conditions is essential, as learning in real world *environments* will typically be limited to a small region of the state space. The ability of the *SANE neurocontroller* to generalise from the unencountered initial conditions is illustrated in figure 5-58 to figure 5-61 and figure 5-62 to figure 5-65. The *neurocontroller* is thus able to robustly generalise to unencountered areas of the state space. This emphasises the *neurocontroller's* ability to learn an effective behavioural *policy* from a small portion of the state space.

The desired *ITAE* response from both initial conditions resulted in transient responses that closely match those of Li & Biegler (1988) (section 5.3.2) in terms of overshoot and settling time. Brengel & Seider (1989) (section 5.3.3) and Gattu & Zafiriou

(1992) (section 5.3.4) were able to significantly reduce the overshoot and settling time for the system. As indicated by Gattu & Zafiriou (1992), the faster obtained dynamics are a function of the weight value selection in the model predictive control algorithm. Gattu & Zafiriou (1992) were able to obtain the same dynamic response as Li & Biegler (1988) and match the response by Brengel & Seider (1989) by adjusting the weight values in the algorithm. The dynamic response is thus a function of tuning parameters and not related to the functionality of the algorithm employed. Requiring *SANE* to find a controller that correlates to the performance setting obtained by Brengel & Seider (1989), requires a modification to the *fitness* function that will result in faster controller dynamics. Instead of the *ITAE* criteria, a criteria that will result in the development of faster controllers would, for example, be -

$$J = \int_0^{\infty} t^2 \cdot |e| \cdot dt \quad (5-12)$$

The function that is integrated thus becomes larger far sooner during a trial, as time progresses, than for the *ITAE* requirement. Minimising equation 5-12 would thus result in faster controller response behaviour.

The desired closed loop response for the concentration of B in the reactor (figure 5-58) from initial condition $C_b = 0.1 \left[\frac{\text{mol}}{\text{dm}^3} \right]$ & $h = 40 \text{ [dm]}$, displays a large degree of overshoot which is not characteristic of the *ITAE* response. This is due the application of the *ITAE* criteria on the reaction rate in *SANE's fitness* function; not on the concentration of B directly. The nonlinear transformation from reaction rate to the concentration of B, results in the loss of *ITAE* characteristics in the response of concentration B. Also, an inverse response occurs in the reaction rate from this initial condition, as the *neurocontroller* attempts to balance the performance criteria for maintain the reactor level and concentration of B. The sharp inflection point is a result of this inverse response and the nonlinear mapping to concentration. The liquid level (figure 5-59) responds in typical *ITAE* fashion, as the *ITAE* criteria is directly applied.

From initial condition $C_b = 7.0 \left[\frac{\text{mol}}{\text{dm}^3} \right]$ & $h = 40 \text{ [dm]}$ (figure 5-62 to figure 5-65), the response has typical *ITAE* characteristics.

This ability to maintain robust performance in the presence of process gain sign change, is a paramount reason for utilising advanced control techniques as linear controllers are ineffective in such process *environments* (section 5.3.1). As may be seen from the two servo responses (figure 5-58 to 5-61 & figure 5-62 to 5-65), the *neurocontroller* is able to effectively drive the process from one side of the gain sign change (unstable steady state ϕ) to the other, without affecting the performance of the closed loop response.

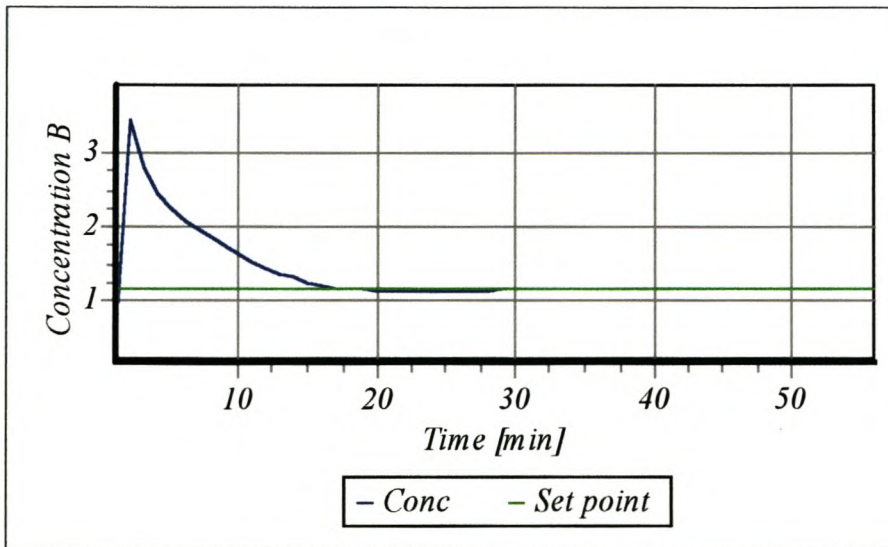


Figure 5-58 - Transient response for concentration B from initial condition $C_b = 0.1$ & $h = 40$.

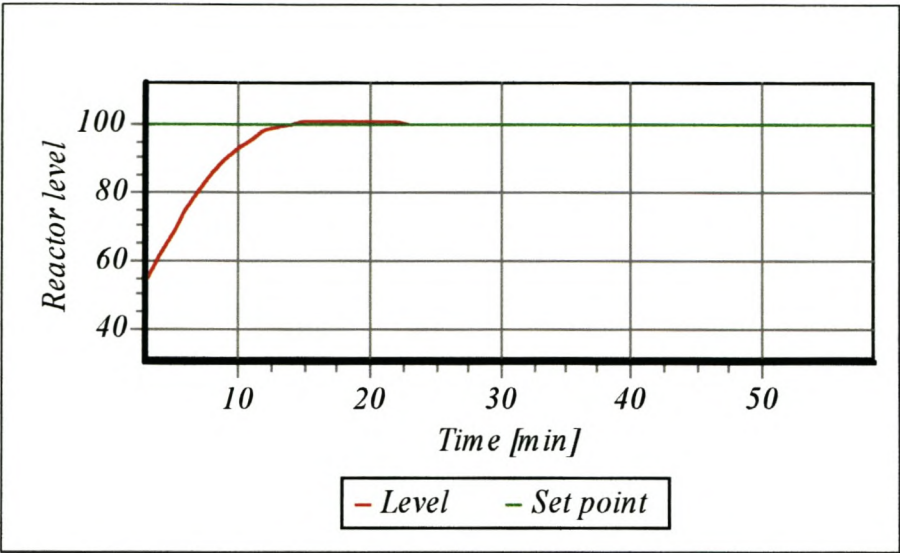


Figure 5-59 - Transient response for the reactor level for initial condition $C_b = 0.1$ & $h = 40$.

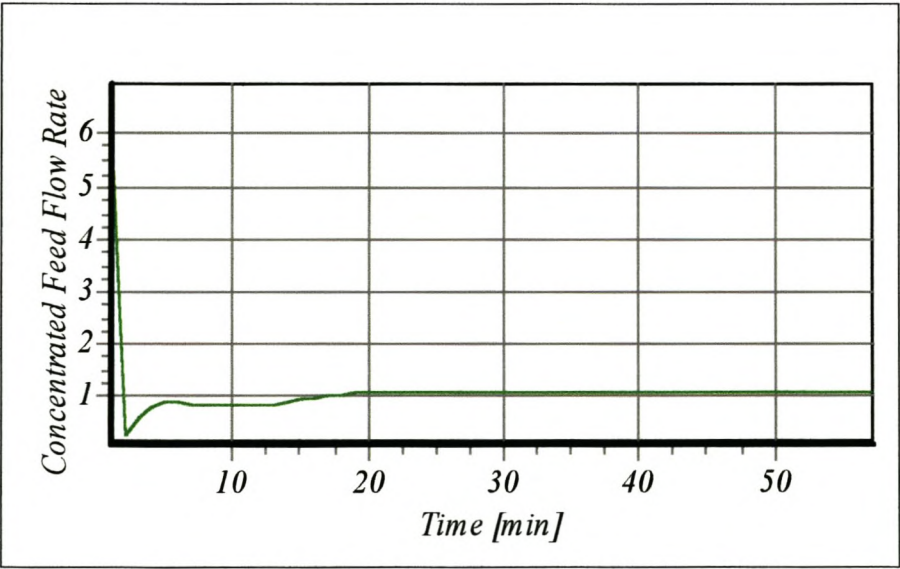


Figure 5-60 - Concentrated feed flow rate control action resulting as a response to initial condition $C_b = 0.1$ & $h = 40$.

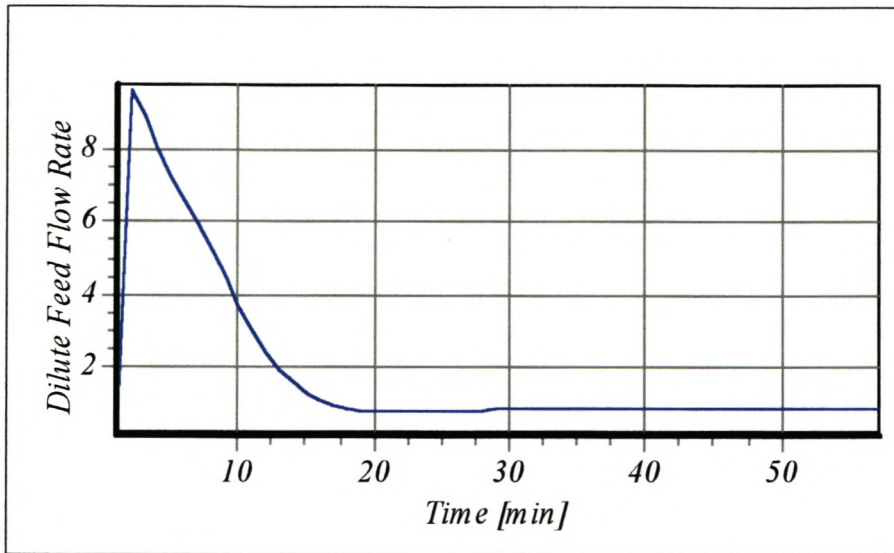


Figure 5-61 - Dilute feed flow rate control action resulting as a response to initial condition $C_b = 0.1$ & $h = 40$.

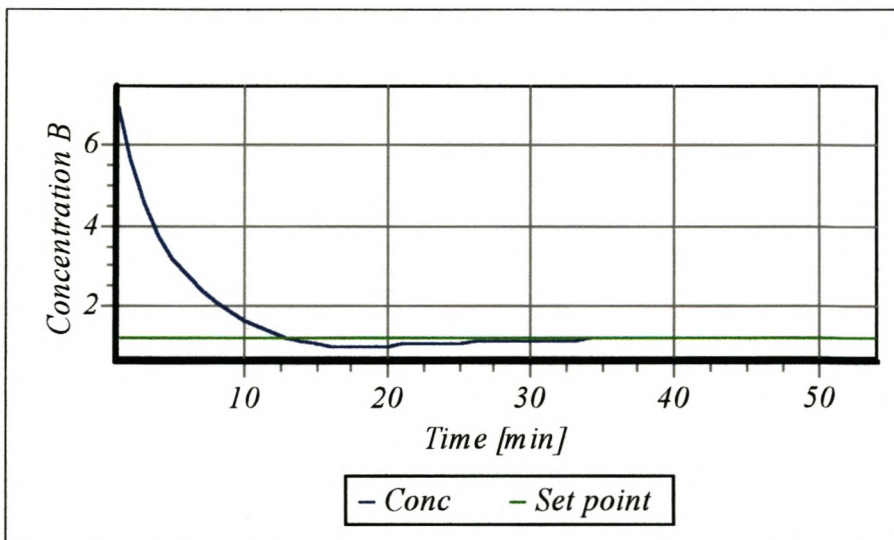


Figure 5-62 - Transient response for concentration B from Initial condition $C_b = 7.0$ & $h = 40$.

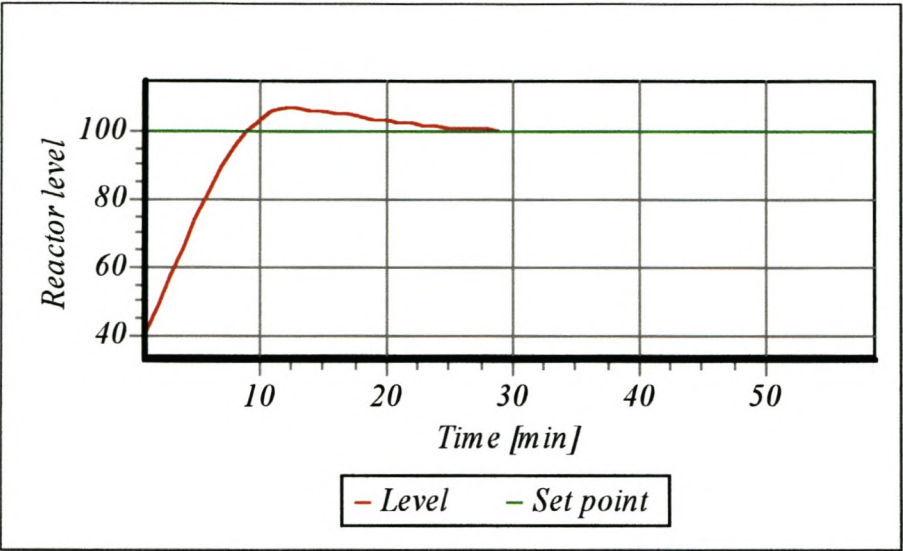


Figure 5-63 - Transient response for the reactor level for initial condition $C_b = 7.0$ & $h = 40$.

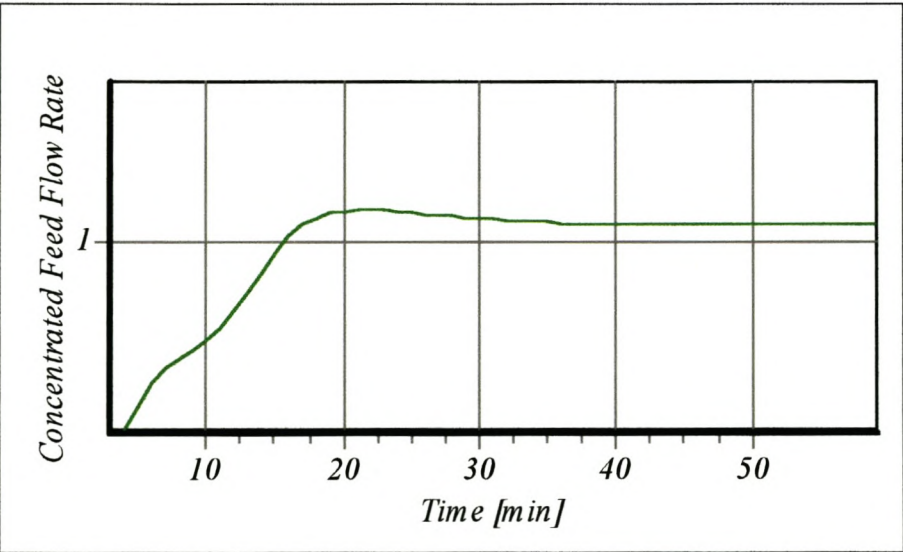


Figure 5-64 - Concentrated feed flow rate control action resulting as a response to initial condition $C_b = 7.0$ & $h = 40$.

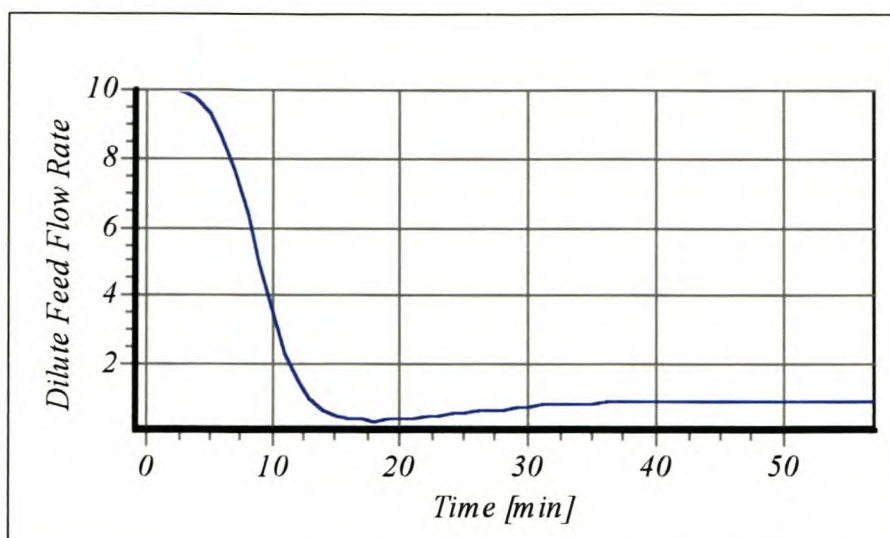


Figure 5-65 - Dilute feed flow rate control action resulting as a response to initial condition $C_b = 7.0$ & $h = 40$.

5.3.6.3 Large Feed Concentration Disturbance

As discussed in section 5.3.1 process disturbances in C_{B0} during open loop operation may lead to operation at point β in figure 5-48, which causes catalyst "wetting", reducing the apparent activation of the catalyst (lower space time yield). Retaining operation at point α involves a complex operation in the open loop (section 5.3.1).

The ability to reject disturbances in the concentrated feed (C_{b1}) is thus critical in maintaining the operation at the high space time yield point α . Brengel & Seider (1989) considered a 20% disturbance in C_{b1} , which resulted in a large *state* as illustrated in figure 5-51. The attained steady state as a result of this disturbance, is located at the stable steady state γ . Without *model mismatch compensation* the controller thus drives the process to stable steady state γ , which is the region of the state space in which catalyst "wetting" dominates the space time yield. Without *model mismatch compensation* in the presence of large feed disturbances, the process thus moves from the unstable steady state to the stable steady state utilising *MPC*. Only once *model mismatch compensation* was incorporated into the model predictive control algorithm, was the process controllable at operating point ϕ as seen in figure 5-51. Smaller disturbances in C_{b1} could, however, be effectively compensated for without *model mismatch compensation*.

The developed *neurocontroller's* robustness in the presence of large disturbance in C_{b1} was considered. A random disturbance with a *gaussian* distribution (mean 24.9 $[\frac{mol}{dm^3}]$, STD 15 $[\frac{mol}{dm^3}]$) was introduced in the concentrated feed. The unstable steady state ϕ is located in near proximity to α for the optimal reaction rate control (*SANE* case in figure 5-47) considered for the *neurocontroller*. The concentrated feed concentration disturbance is illustrated in figure 5-66, which frequently corresponds to a disturbance greater than 20%. Figure 5-67 illustrates the open loop response to this disturbance, with constant flow rates that would maintain the set point should no disturbances be present (open loop stable). The introduced feed disturbance drives the reactor to operate in the region of γ . The close proximity of ϕ to α , results in γ being more stable than α from an operation's viewpoint. The transient response to this disturbance in the closed loop from an initial condition of $C_b = 3.31 [\frac{mol}{dm^3}]$ and $h = 68.85 [dm]$, is illustrated in figure 5-68 to figure 5-71. As illustrated in figure 5-67 the desired high space time yield set point (highest economic benefit) is maintained effectively, despite the presence of large disturbances in C_{b1} .

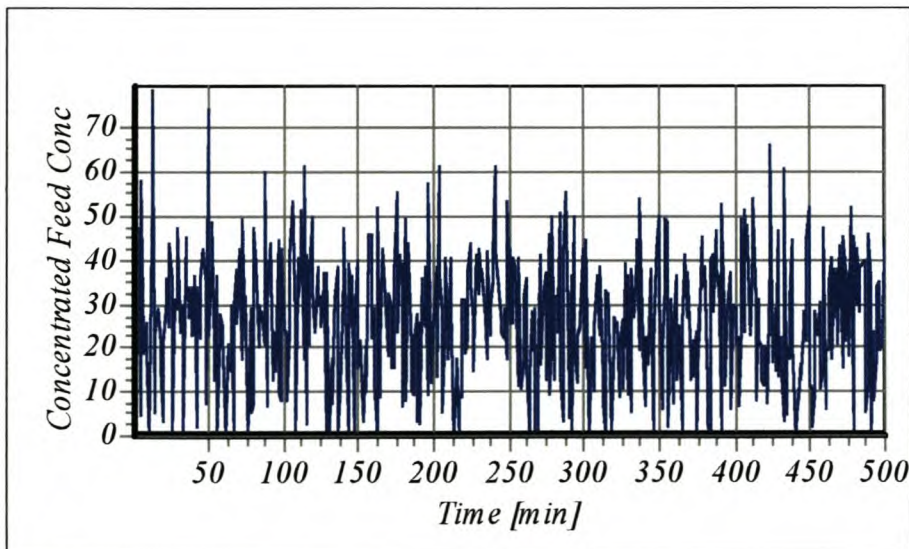


Figure 5-66 - Concentrated feed disturbance with a gaussian distribution around the mean of $C_{b1} = 24$, with a standard deviation of 15.

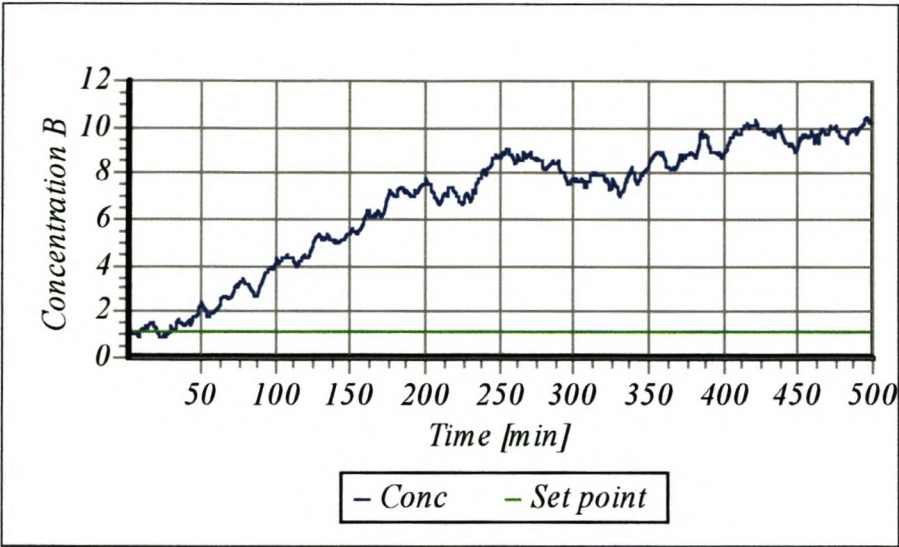


Figure 5-67 - Open loop response from initial condition $h = 100$ & $C_b = 1.163$, with large disturbances in the inlet feed concentration (C_{b1}). $w_1 = 1.087$ and $w_2 = 0.9145$.

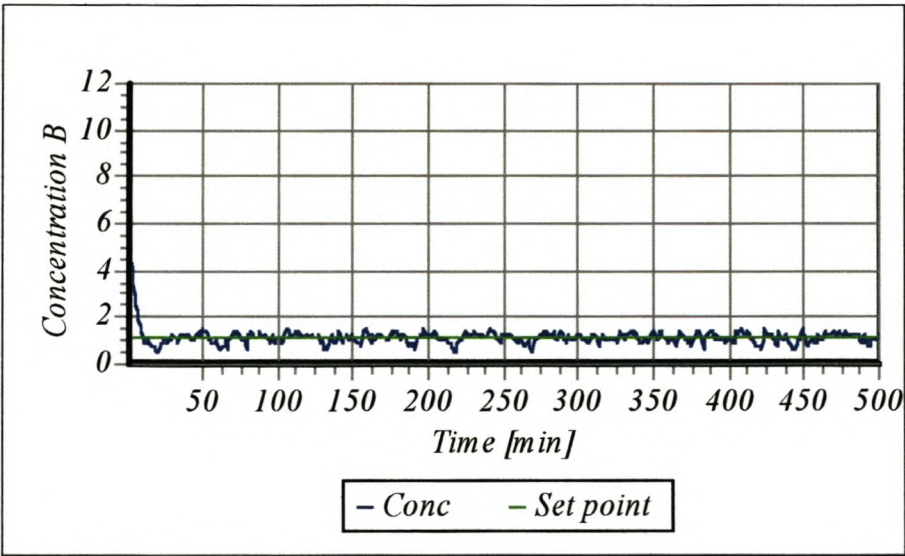


Figure 5-68 - Closed loop response for concentration B from initial condition $C_b = 3.31$ and $h = 68.85$.

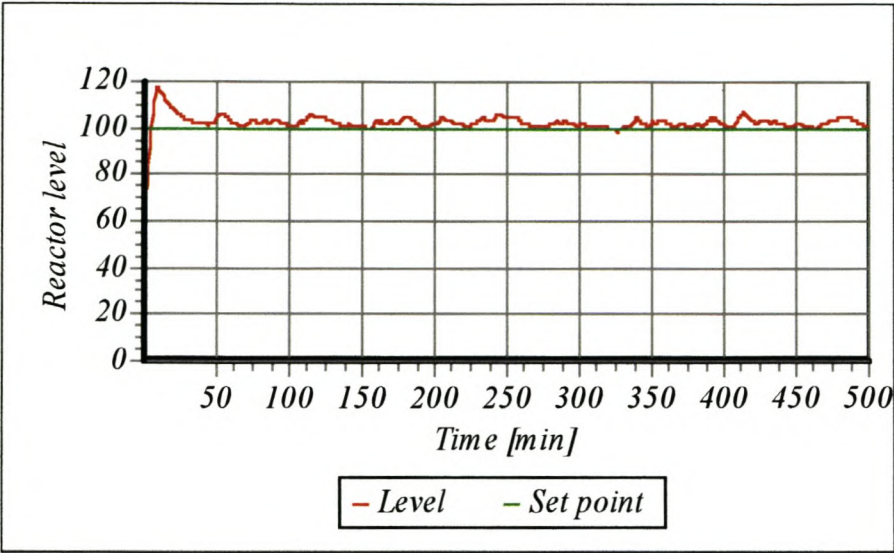


Figure 5-69 - Closed loop response for Reactor level from initial condition $C_b = 3.31$ and $h = 68.85$.

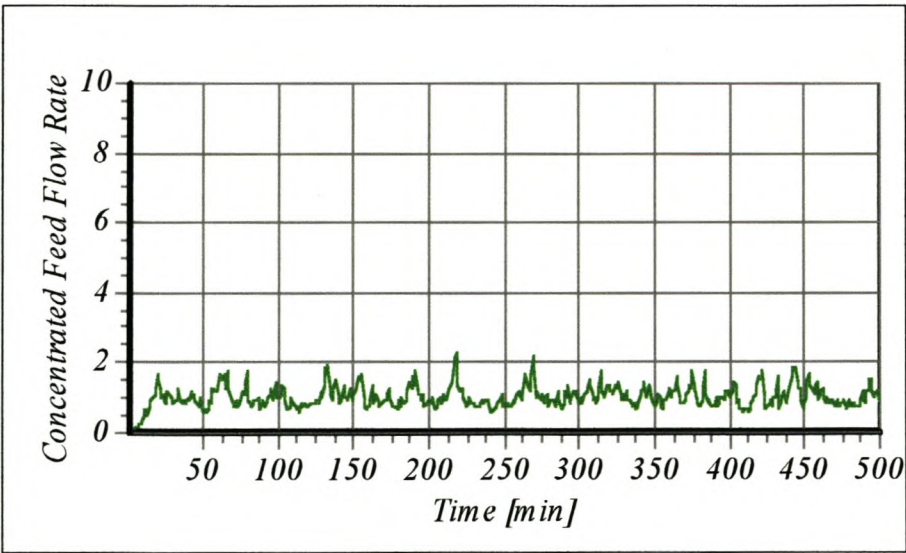


Figure 5-70 - Concentrated feed flow rate control action to an initial condition $C_b = 3.31$ and $h = 68.85$.

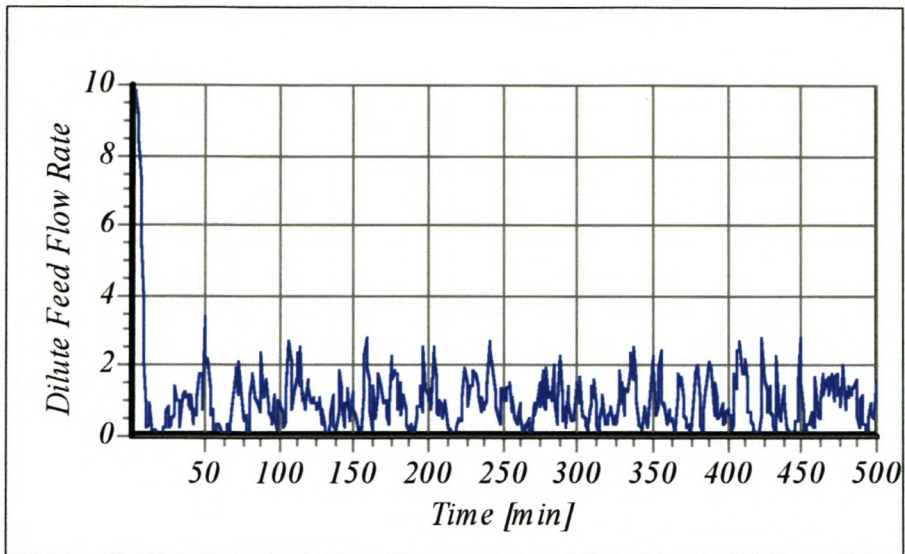
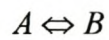


Figure 5-71 - Dilute feed flow rate control action to an initial condition $C_b = 3.31$ and $h = 68.85$.

5.4 NONISOTHERMAL REACTOR SIMULATION

5.4.1 Nonisothermal Reactor Process Description

Li & Biegler (1988) considered a first-order reversible exothermic reaction system



(5-13)

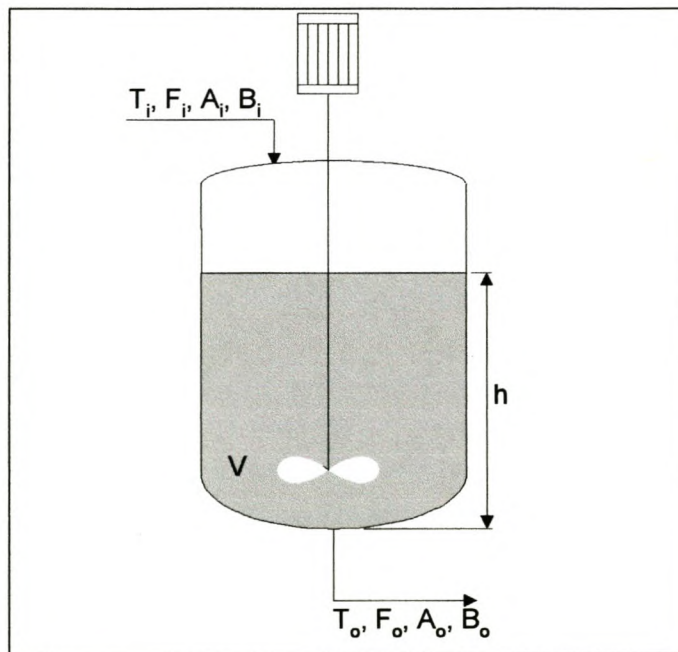


Figure 5-72 - Reversible Stirred tank reactor

that is carried out in an ideal stirrer tank reactor as shown in figure 5-72. The combined concentration of A and B is assumed constant. The tank is uniformly mixed with the outlet flow rate being determined by the liquid height (head) in the reactor.

$$F_0(h) = 2.5 \cdot h^{0.5} \quad (5-14)$$

The nonlinear differential-algebraic equations that model the dynamics of the reactor are derived from differential mass and energy balances. After numerical value substitution, the model, with model parameters described in table 5-4, may be simplified to -

$$\frac{dx_1}{dt} = -0.16 \cdot \frac{x_1}{x_3} \cdot u_1 + K_1 \cdot (1 - x_1) - K_2 \cdot x_1 \quad (5-15)$$

$$\frac{dx_2}{dt} = 0.16 \cdot \frac{x_2}{x_3} \cdot u_2 - 0.16 \cdot \frac{x_2}{x_3} \cdot u_1 + 5 \cdot [K_1 \cdot (1 - x_1) - K_2 \cdot x_1] \quad (5-16)$$

$$\frac{dx_3}{dt} = 0.16 \cdot u_1 - 0.4 \cdot x_3^{0.5} \quad (5-17)$$

$$K_1 = k_{01} \cdot e^{\left(\frac{E_1}{x_2}\right)} \quad (5-18)$$

$$K_2 = k_{02} \cdot e^{\left(\frac{E_2}{x_2}\right)} \quad (5-19)$$

Table 5-4 - Model parameter description and formulation for the nonisothermal reactor

<i>Description</i>	<i>Formulation</i>	<i>Unit</i>
Conversion of reactant A	x_1	[]
Reactor outlet temperature	x_2	[K]
Reactor Level	x_3	[K]
Inlet flow rate	u_1	$[\frac{1}{\text{min}}]$
Feed inlet temperature	u_2	[K]
Forward reaction activation energy	$E_1 = 5000$	[K]
Reverse reaction activation energy	$E_2 = 7500$	[K]
Forward reaction Arrhenius constant	$k_{01} = 3 \cdot 10^5$	$[\text{min}^{-1}]$
Reverse reaction Arrhenius constant	$k_{02} = 6 \cdot 10^7$	$[\text{min}^{-1}]$

Analysis of the steady state equations reveals that, for a fixed inlet flow rate (constant level), the reactor equilibrium conversion as a function of inlet flow temperature has a well defined maximum (figure 5-73).

The control objective is to operate the reactor as closely as possible to this optimal point subject to the process constraints, maintaining the stability of the closed loop system in the face of disturbances and/or unmodelled system dynamics. Figure 5-73 illustrates the equilibrium conversion vs. reaction temperature (x_2) when the height of the liquid level (y_1) is at the set point ($y_1^{sp} = 0.16[m]$). The *manipulated variables* are the inlet flow rate and the inlet temperature to the reactor.

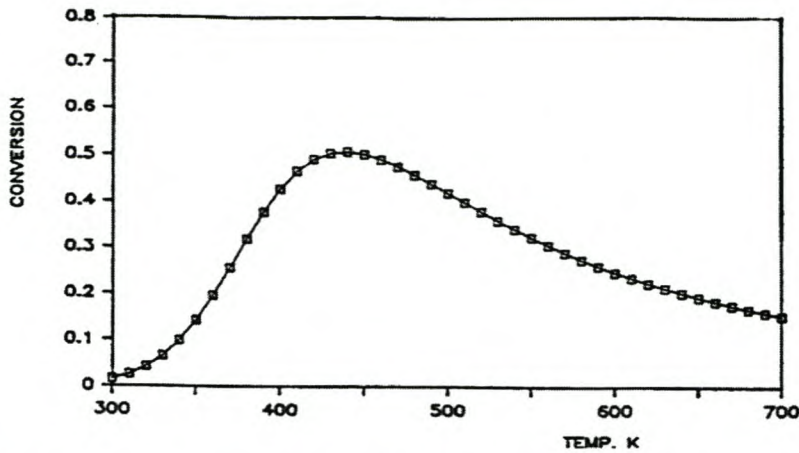


Figure 5-73 - Equilibrium diagram for the $h = 0.16[m]$ (Economou & Morari, 1986).

Due to the conversion maximum, the system gain changes sign as the operating point changes from one side of the maximum to the other. Systems of this type are not "integral controllable". Such systems with integral control cannot be stable over the entire operating region when controllers with integral action are employed. For stability to be achievable, the "no offset requirement" needs to be abandoned when linear controllers are utilised. Under proportional feedback when a temporal disturbance of -2% is introduced in the inlet reactant concentration, the linear controller pushes the system beyond the optimal operating temperature and instability follows. A state disturbance that drives the process beyond T_{opt} was also considered. The linear controller was also unable to recover the system; operating the system at unnecessarily high temperatures with offset (Economou & Morari, 1986).

Of significance to control structures that utilise the system inverse, the steady-state input-output mapping is not invertable. For a given output conversion there are two values of the *manipulated variable* that can produce the desired output (a low and a high temperature).

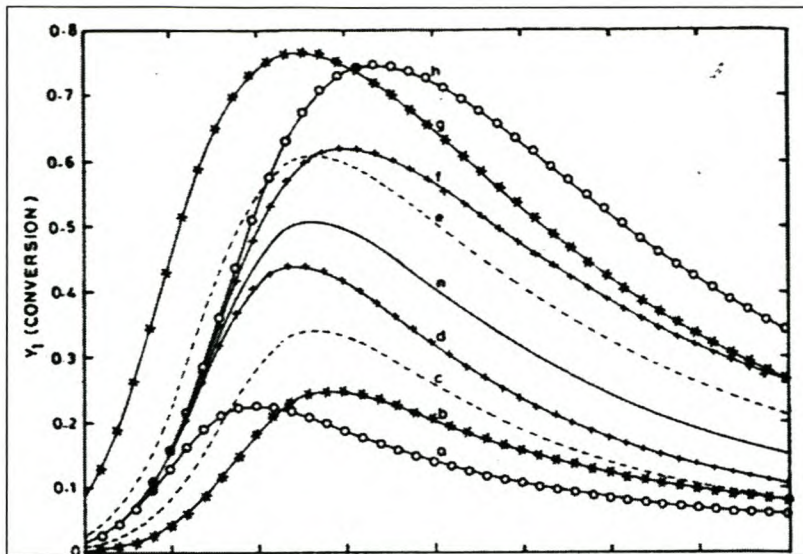


Figure 5-74 - Reactor steady-state diagrams for nominal model and perturbations in the kinetic parameters of the plant. (n) nominal model, (a) -10% E_2 , (h) +10% E_2 , (b) +10% E_1 , (g) -10% E_1 , (c) -50% K_{01} , (e) +50% K_{01} , (d) +50% K_{02} , (f) -50% K_{02} (Patwardhan & Madhavan, 1993).

5.4.2 Single Step pseudo-Newton Model Predictive Control algorithm

Li & Biegler (1988) used two sets of initial conditions to demonstrate their one-step algorithm. One initial condition is left of the maximum point of the equilibrium reactor curve; the other is to the right of the maximum point (table 5-5). Significantly, the linear *IMC* is unstable at the second set of initial conditions with fixed height of the liquid level (Economou & Morari, 1986).

Table 5-5 - Initial conditions considered by Li & Biegler (1988)

Initial condition Set 1	Initial condition Set 2
$x_1 = 0.16 []$	$x_1 = 0.41 []$
$x_2 = 351 [K]$	$x_2 = 503 [K]$
$x_3 = 0.12 [m]$	$x_3 = 0.20 [m]$
$u_1 = 0.866 [\frac{\ell}{\min}]$	$u_1 = 1.12 [\frac{\ell}{\min}]$
$u_2 = 352 [\frac{\ell}{\min}]$	$u_2 = 504 [\frac{\ell}{\min}]$

Li & Biegler (1988) observed that the most satisfactory controller response was obtained for their algorithm, when the upper bound on u_2 is 490 [K]. The responses for the single step pseudo-Newton algorithm are illustrated in section 5.4.3 along with the results by Patwardhan & Madhavan (1993).

5.4.3 Nonlinear Model Predictive Control Using Second-Order Model Approximation

5.4.3.1 Servo characteristics

Patwardhan & Madhavan (1993) similarly considered the reactor proposed by Li & Biegler (1988) (section 5.4.2). Two versions of the nonlinear *MPC* algorithm were considered: (1) the *NLMPC1* algorithm which utilises a locally linear prediction model; (2) the *NLMPC2* algorithm using a second-order prediction model.

The following process constraints were imposed -

$$0 \leq u_1 \leq 2.5 \quad \left[\frac{1}{\text{min}} \right] \quad (5-20)$$

$$300 \leq u_2 \leq 490 \quad [\text{K}] \quad (5-21)$$

The simulation results, under the perfect model assumption (no model-plant mismatch) are presented in figure 5-75 to figure 5-79. From figure 5-75 it is evident that the *NLMPC2* algorithm forces the controlled *process variables* to approach their set points along smoother trajectories, also resulting in faster dynamics. The extent of the control action on the inlet temperature is also less for the *NLMPC2* algorithm, than is the case for Li & Biegler's (1988) algorithm (figure 5-75). Patwardhan & Madhavan (1993) attribute the superior performance of *NLMPC2* to a judicious choice of weights on errors in concentration and level in the objective function.

For the conversion transient response, no significant difference exists when the single step *NLMPC1* or the *NLMPC2* algorithms are used (figure 5-75). The two single step algorithms, however, show significant difference in the temperature response (figure 5-76). While *NLMPC2* moves the reactor temperature to the required steady state

without overshoot or oscillation, the NLMPC1 algorithm produces an oscillatory temperature response. The oscillatory response in reactor temperature is due to the large oscillations in the control action of u_2 produced by NLMPC1 (figure 5-79). This is a result of the linear prediction model used in NLMPC1 being unable to predict the change in sign of the gain in the vicinity of the optimum setpoint. Also, low values of sensitivity to small inlet flow temperature changes contribute to the observed oscillatory behaviour in the reactor inlet temperature. A multi-step NLMPC1 algorithm ($P = 10$) was found to stabilise the temperature response, at the expense of a slower response in approach to the conversion set point (Patwardhan & Madhavan, 1993).

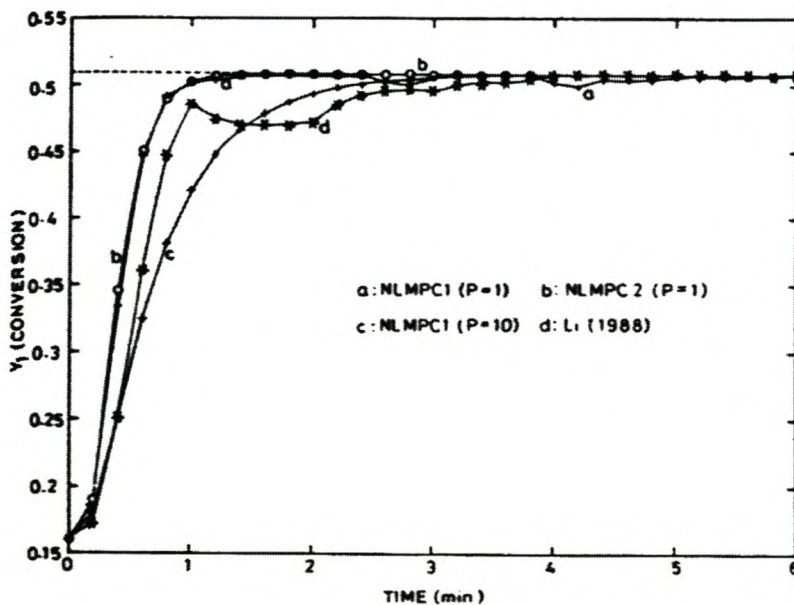


Figure 5-75 - Conversion transient response for the NLMPC1, NLMPC2 and Li & Biegler (1988); (Patwardhan & Madhavan, 1993).

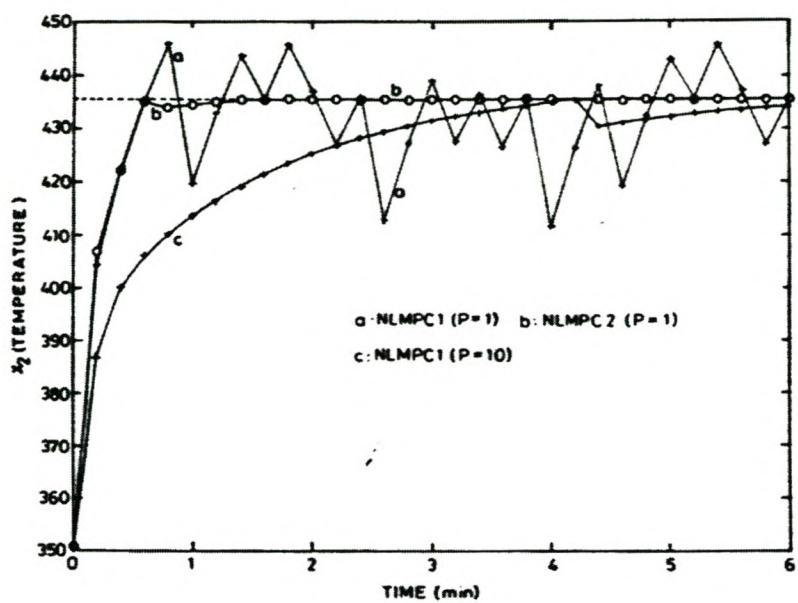


Figure 5-76 - Reactor outlet temperature transient response for NLMPC1 and NLMPC2 (Patwardhan & Madhavan, 1993).

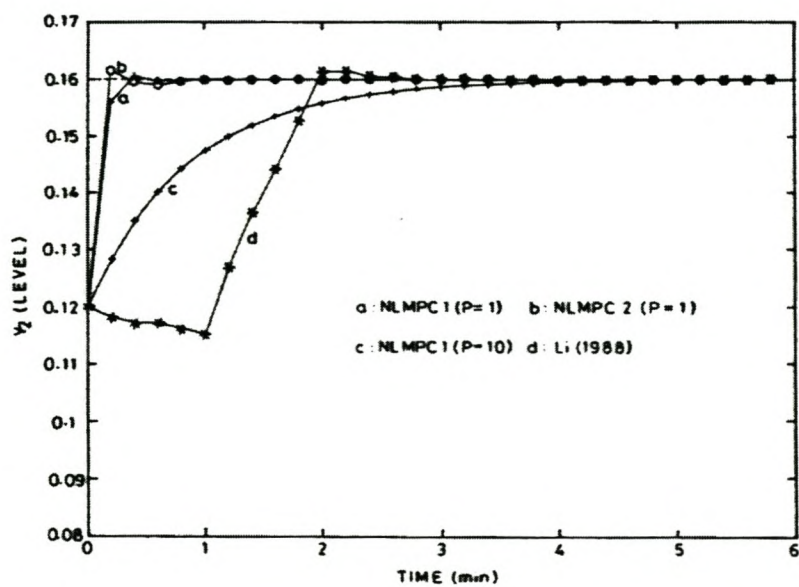


Figure 5-77 - Reactor level transient response for NLMPC1, NLMPC2 and Li & Biegler (1988); (Patwardhan & Madhavan, 1993).

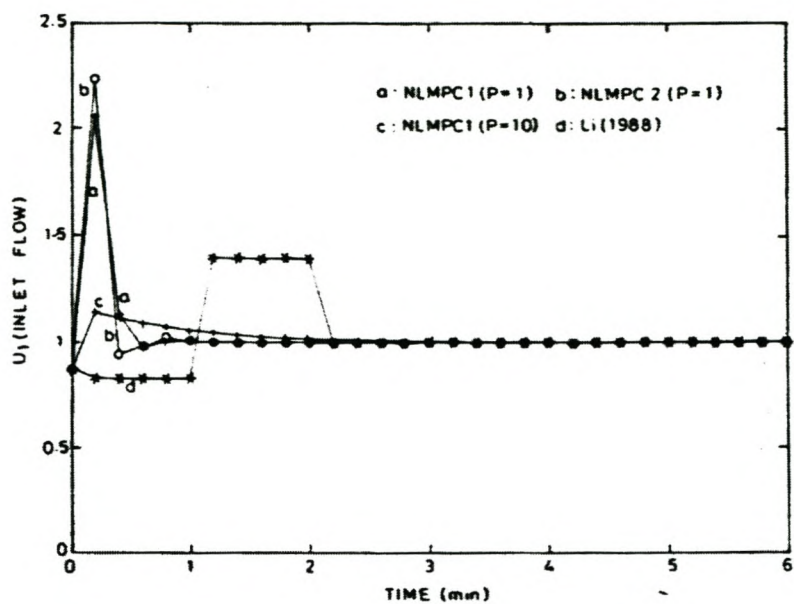


Figure 5-78 - Flow rate into reactor for NL MPC1, NL MPC2 and Li & Biegler (1988); (Patwardhan & Madhavan, 1993).

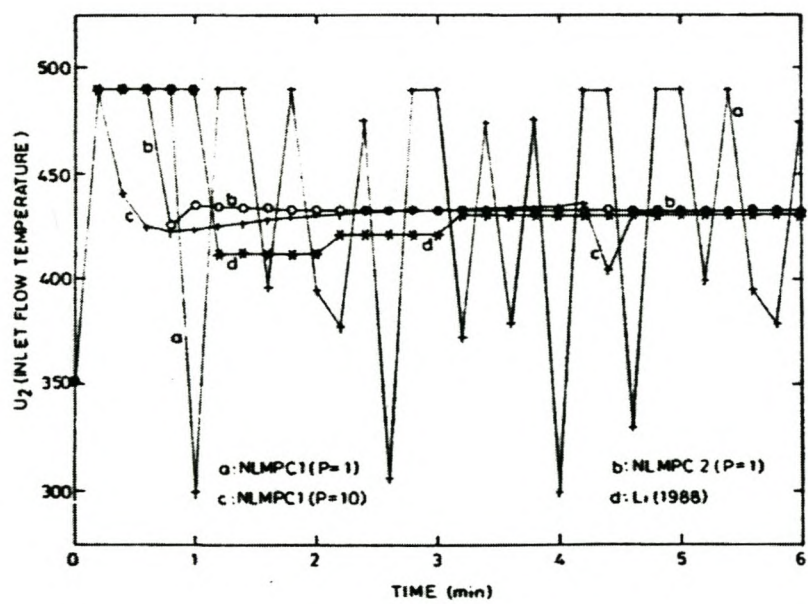


Figure 5-79 - Reactor inlet temperature for NL MPC1, NL MPC2 and Li & Biegler (1988); (Patwardhan & Madhavan, 1993).

5.4.3.2 Model / Plant mismatch analysis

To ensure robustness of the control algorithms, *NLMPC* and *NLMPC2*, the effect of perturbations in the kinetic parameters of the plant was considered ($\pm 10\%$ variations in E_1 and E_2). The location of the steady state maximum conversion as a function of these uncertainties is marked (figure 5-74) (Patwardhan & Madhavan, 1993).

When the parameter perturbation (+10% uncertainty in E_1) shifts the maximum achievable conversion in the plant below the maximum conversion predicted by the nominal model (figure 5-80), the conversion set point (based on the nominal model optimum) cannot be achieved, resulting in *state* (figure 5-80a). The mismatch correction compensation term has the effect of only shifting the origin of the linear and quadratic perturbation model, while preserving the geometric characteristics. Both *NLMPC1* and *NLMPC2* attempt to maximise the conversion by moving toward the nominal model peak, which in this perturbation model is not attainable. The maximum attainable conversion with the +10% perturbation in E_1 is illustrated in figure 5-74, which is attained by both the *NLMPC1* and *NLMPC2* algorithms. The single step *NLMPC2* algorithm maintains a stable non-oscillatory *manipulated variable* u_2 profile, while the single and multi-step *NLMPC1* lead to oscillatory responses (figure 5-80b). The oscillatory response for *NLMPC1* is not damped by extending the prediction horizon (Patwardhan & Madhavan, 1993).

When the parameter perturbation (-10% E_1) shifts the maximum achievable conversion in the plant above the maximum conversion predicted by the nominal model (figure 5-81a), the conversion set point can be reached at an operating temperature away from the temperature at the nominal optimum peak. In this region the nominal model has a dominant linear component, which may influence the closed loop response. Both single step *NLMPC1* and *NLMPC2* give rise to oscillations in the conversion transient and the *manipulated variable* u_2 response. The use of multi-step *NLMPC1* and *NLMPC2* algorithms eliminates these oscillations (figure 5-81b).

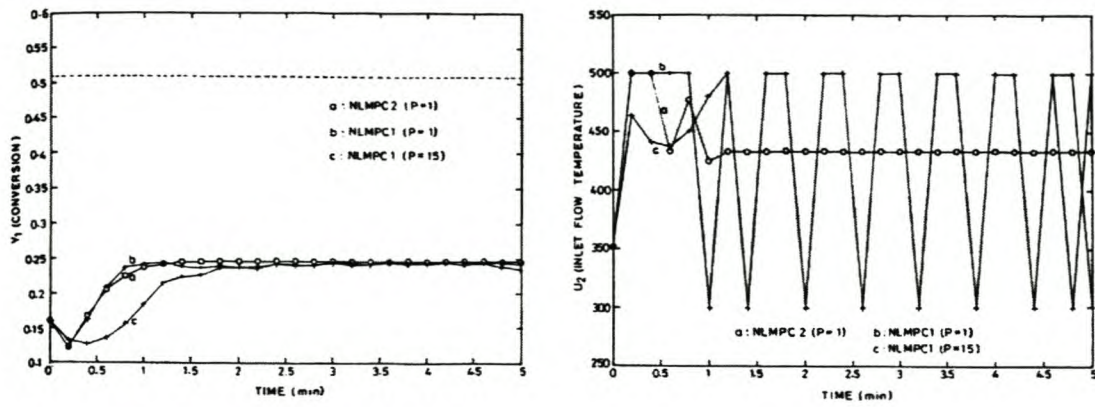


Figure 5-80 - +10% perturbation in E_1 (Patwardhan & Madhavan, 1993).

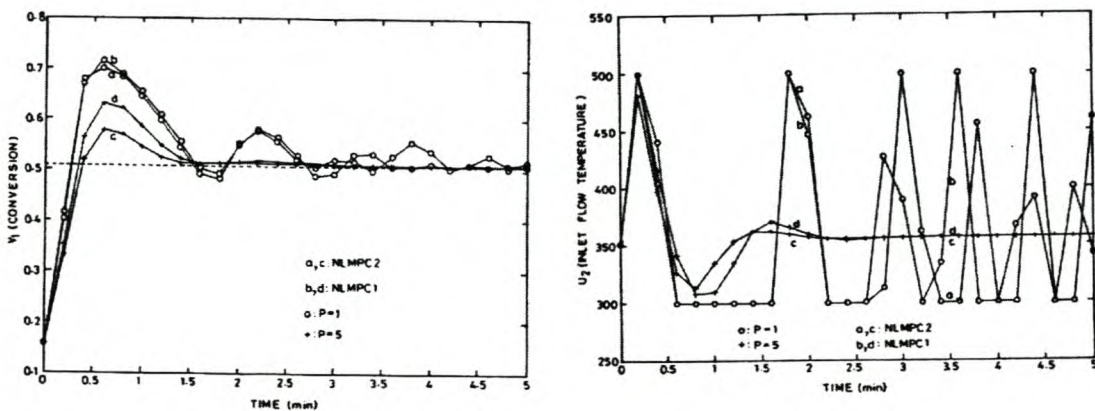


Figure 5-81 - Perturbation in E_1 (-10%) (Patwardhan & Madhavan, 1993).

5.4.4 Direct and Indirect Model Based Control using ANN

Psichogios & Ungar (1991) investigated the nonisothermal reactor system with a constant reactor volume (*SISO* system). The objective set by Psichogios & Ungar (1991) was to demonstrate both direct and indirect methods for incorporating neural networks into the control structure. As the system is not invertible, the use of neural network inverses is problematic. Consequently, the input-output examples used to train the *networks* were restricted to a region in the state space where the inverse mapping is unique.

A configuration of four input nodes, a single hidden layer with eight *neurons* and an output layer of three output *neurons* was used to model the forward system dynamics. The inputs were the three state variables and the *manipulated variable* (inlet temperature). A second *network* was trained to represent the inverse dynamics of the

process. The inverse *network* configuration consisted of four input nodes; the three state variables and the target value of the *process variable* (reactor temperature) for the next sampling instant. A single output node specified the value of the *manipulated variable* (inlet temperature) for the current sampling interval (Psichogios & Ungar, 1991).

Both the neural *networks* were incorporated into the *IMC* structure (section 4.1.4) as the process model and the controller. Following this approach, the control action at each sampling interval was the output of the trained inverse model *network*. The proposed dual *network* control structure was unable to compensate for a +2% disturbance in the inlet concentration of B. This lack of robustness was attributed to learning difficulties. Using neural *networks* to learn both the forward and the inverse process dynamics involved an approximation error, which is typically different for the two training episodes. In this case, the inverse is also a particular approximation to the actual process inverse. The poor performance of the *networks* in the *IMC* structure is attributed to the approximation errors involved in learning the inverse dynamics (Psichogios & Ungar, 1991).

A direct inverse control implementation (section 4.1.2) without a filter resulted in some *state*. A similar implementation with a filter eliminated the offset, but resulted in a very sluggish response. Psichogios & Ungar (1991) noted that in practice, such a direct inverse controller would have to be made robust (e.g. including a linear controller in the feedback loop), due to the approximation error in modelling the inverse dynamics. Failing to make the control structure robust, may result in unstable closed loop operation or significant offset.

An alternative inverse model implementation involved calculating the inverse on-line from the *network* describing the process dynamics, thus constructing a rigorous inverse process model. This indirect *IMC* type controller with "detuning", was found to have good response disturbance rejection characteristics and no steady-state offset.

Psichogios & Ungar (1991) also presented a model predictive controller implementation (section 4.1.5). The *MPC* implementation was found to exhibit some steady-state offset. This offset was attributed to the inherent modelling error involved

in training the process dynamics *network* and the fact that the *MPC* approach is sensitive to modelling error. Incorporating *model mismatch compensation* resulted in minimal *state*.

Psichogios & Ungar (1991) concluded from the above analysis that modelling errors typically present when using neural *networks* as both the process models or as inverse models, require that both direct and indirect controllers be "detuned" to achieve good performance. This "detuning" may involve the incorporation of filters or classical feedback control elements. From a direct control perspective robustness issues exist when neural *networks* are used as controllers, which may be attenuated by an inappropriate choice of control architecture (i.e., *IMC*). It was also concluded that the indirect control approach (i.e., model predictive control) offers more flexibility and generally offers superior performance.

5.4.5 Concluding remarks for control strategies considered in previous research

The nonisothermal reactor is valuable as a control algorithm *evaluation* benchmark, as the process model is not invertible around the operating point and the process exhibits a process gain sign change around the optimum operating temperature. Such processes are not integral controllable, which demands the use of advanced control techniques.

The model predictive control implementations displayed a sensitivity to model/plant mismatch errors. Careful selection of the prediction model needed to be made, in order to obtain satisfactory performance for perturbations in the reaction activation energies. This may require an iterative approach to robust controller design.

The direct and indirect neural *network* control implementations for this problem are of particular interest, as the selection of an appropriate control structure in which to incorporate the neural *networks* is paramount to obtaining satisfactory performance. As the model is not invertible at the optimal operating point, the determination of a

satisfactorily performing inverse control *network* required detuning of the inverse model utilising linear filters or classical control elements.

5.4.6 Symbiotic, Adaptive Neuro Evolution (SANE)

The *SANE* algorithm was utilised to train a *neurocontroller* with five input nodes, 12 hidden nodes and 2 output nodes. The input nodes incorporated the three state variables, and two set point specifications for the reactant conversion and the liquid level. The 2 output nodes provide the outputs for the two *manipulated variables*, the flow rate into the reactor and the inlet temperature. As in section 5.4.2 and 5.4.3 the *MIMO* system is considered for control *evaluation*.

5.4.6.1 Neurocontroller evaluation learning

As in section 5.4.5, *SANE* was required to learn to control the process over a small bandwidth of initial conditions. This is consistent with a real world scenario, as only sparse *domain* information is typically available to the *reinforcement learning* algorithm from which to learn a general behaviour *policy*. For the developed *reinforcement learning agent* to be effective in the *environment*, the *agent* would need to generalise to states or unexpected events not encountered during learning. A typical response of the best *neurocontroller* from an initial condition within the *domain* of learning is illustrated in figure 5-82 to figure 5-85. As this represents learned experience, the controller would be expected to have high performance.

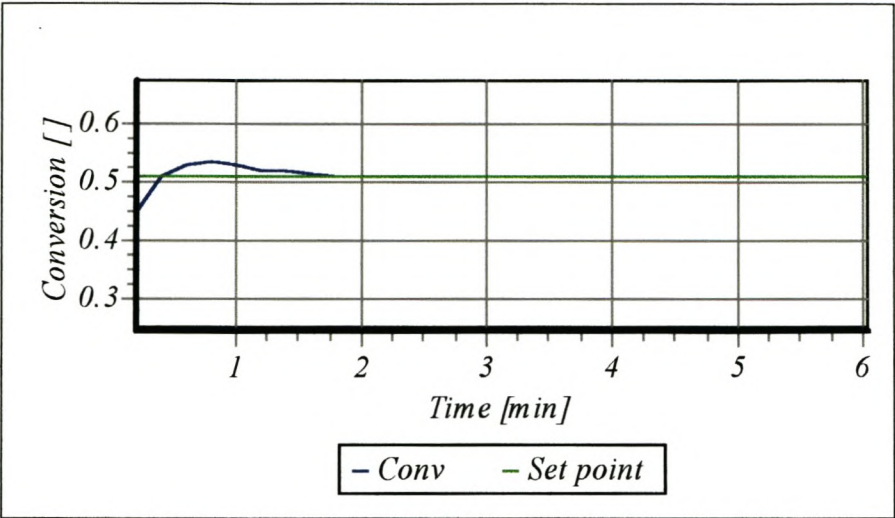


Figure 5-82 - Conversion transient response from initial condition $x_{10} = 0.29$, $x_{20} = 436.98$ K and $h = 0.22$

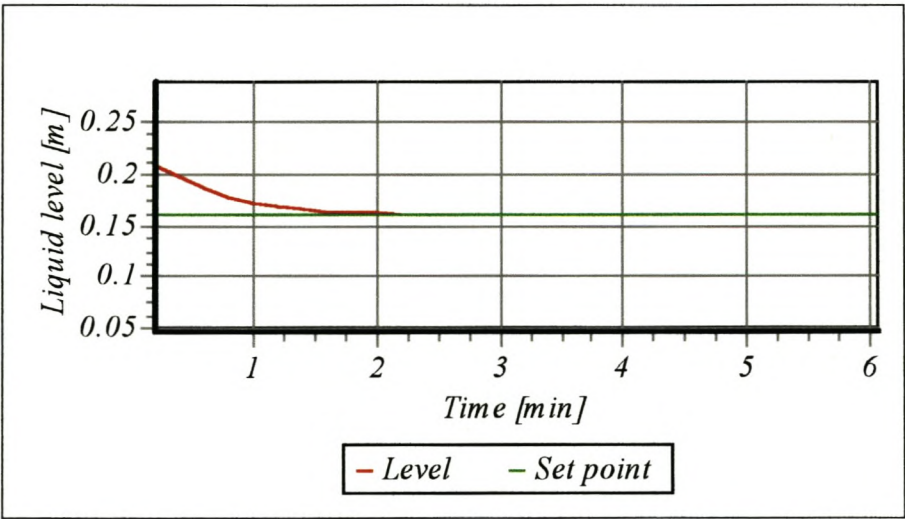


Figure 5-83 - Liquid level transient response from initial condition $x_{10} = 0.29$, $x_{20} = 436.98$ K and $h = 0.22$

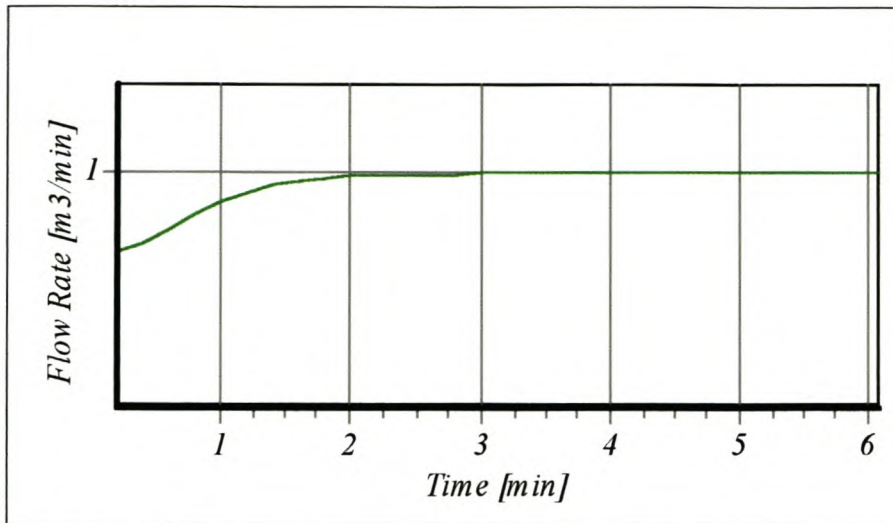


Figure 5-84 - Flow rate control action for initial condition $x_{10} = 0.29$, $x_{20} = 436.98$ K and $h = 0.22$

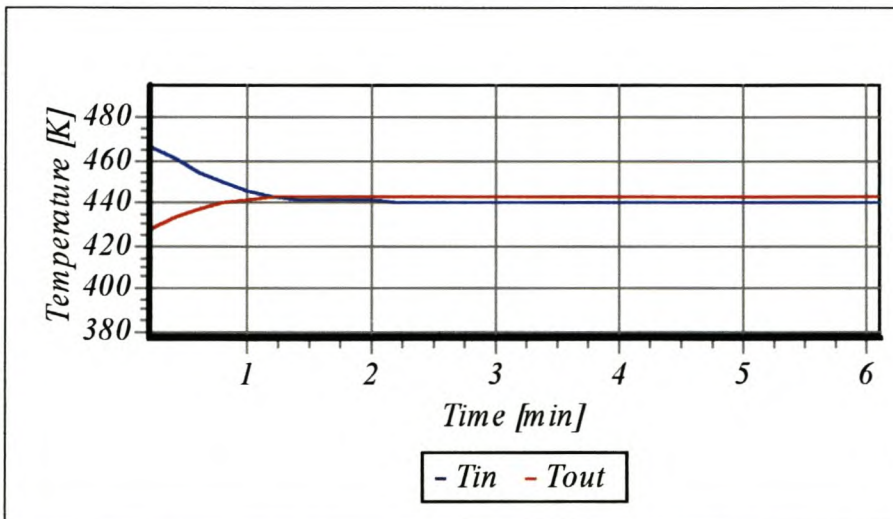


Figure 5-85 - Inlet temperature control action and reactor temperature outlet transient responses. Initial condition $x_{10} = 0.29$, $x_{20} = 436.98$ K and $h = 0.22$.

5.4.6.2 Servo Response Characteristics

Two initial conditions, at either side of the optimum point, are presented by Li & Biegler (table 5-5) to evaluate the servo characteristics of the *neurocontroller*. For a single controller to show effective performance on either side of the optimum conversion, the controller must be able to compensate for the process gain sign change across the optimum conversion point (section 5.4.1). Also, the controller must be able to reach the desired optimum conversion from any initial starting condition.

The initial reactor outlet temperature during learning was randomly generated with a *gaussian* distribution around the optimum conversion temperature, with a standard deviation of 20K. The initial reactor outlet temperature in both sets of initial conditions in table 5-5, fall outside the state space region in which the *neurocontroller* learned to control the process. The transient response from each initial condition state is illustrated in figure 5-86 to figure 5-89 for *set 1* and figure 5-90 to figure 5-93 for *set 2*. The *neurocontroller* is thus able to generalise to regions of the state space not encountered during learning. The *neurocontroller* is able to maintain robust performance in the presence of a process gain sign change.

When comparing figure 5-86 to figure 5-89 to the servo characteristics presented in section 5.4.3, the *neurocontroller* exhibited no oscillatory response in either state variables or *manipulated variables*, as found for the one step locally linearised *MPC* implementation. The *neurocontroller* transient response adhered to the *ITAE* characteristics, which represented a faster settling time than for the 10 step *NLMPC1* implementation, but a slower settling (more damped) response than the *NLMPC2* implementation. As indicated in section 5.4.3 the response characteristics for the *NLMPC* implementations, were a result of judicious selection of algorithm weighting parameters. The settling time of the *SANE* implementation may be expected to improve, with minimal overshoot, for a more aggressive *fitness* criteria related to plant dynamics.

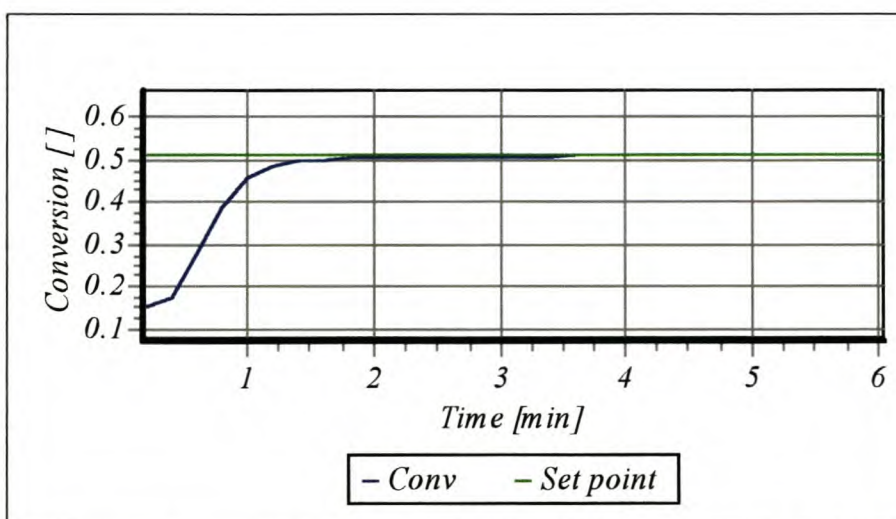


Figure 5-86 - Reactant conversion transient response from initial condition set 1 (section 6.6.1)

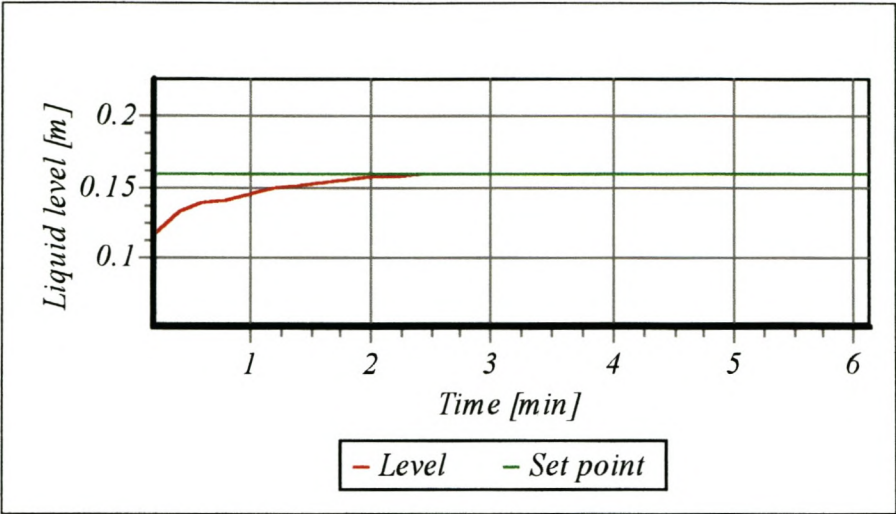


Figure 5-87 - Liquid level transient response form initial condition set 1 (section 6.6.1)

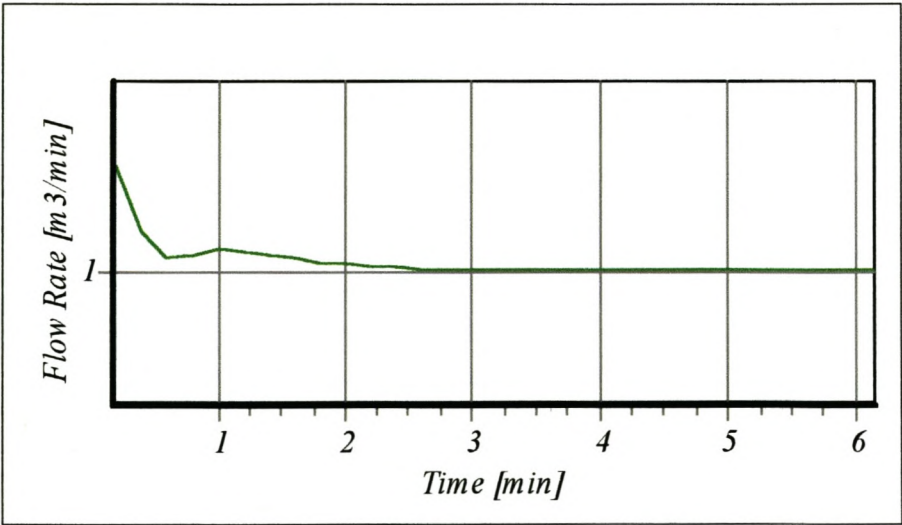


Figure 5-88 - Flow rate control action for initial condition set 1 (section 6.6.1)

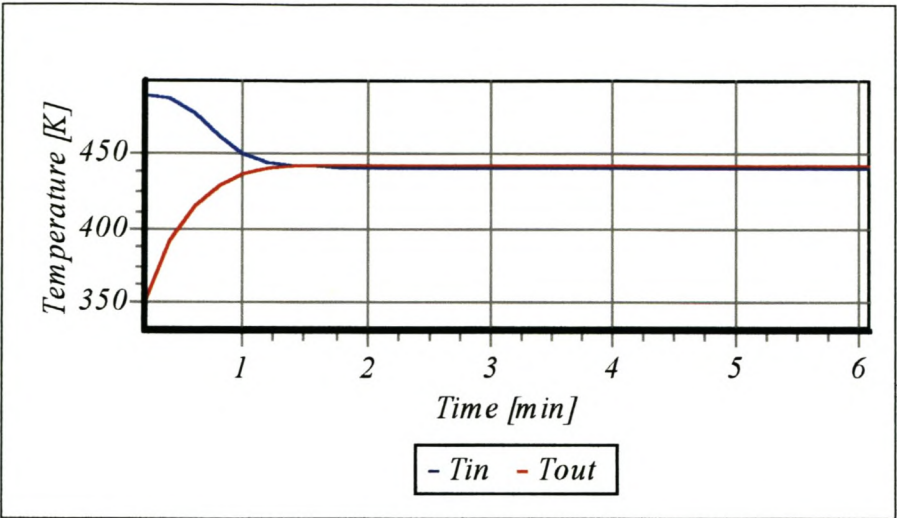


Figure 5-89 - Inlet temperature control action and reactor outlet temperature transient response. Initial condition set 1 (section 6.6.1)

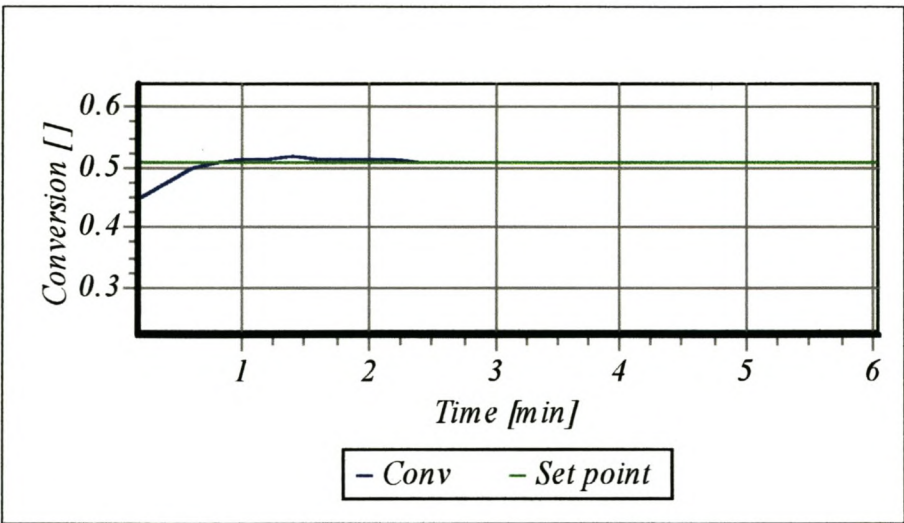


Figure 5-90 - Reactant conversion transient response from initial condition set 2 (section 6.6.1)

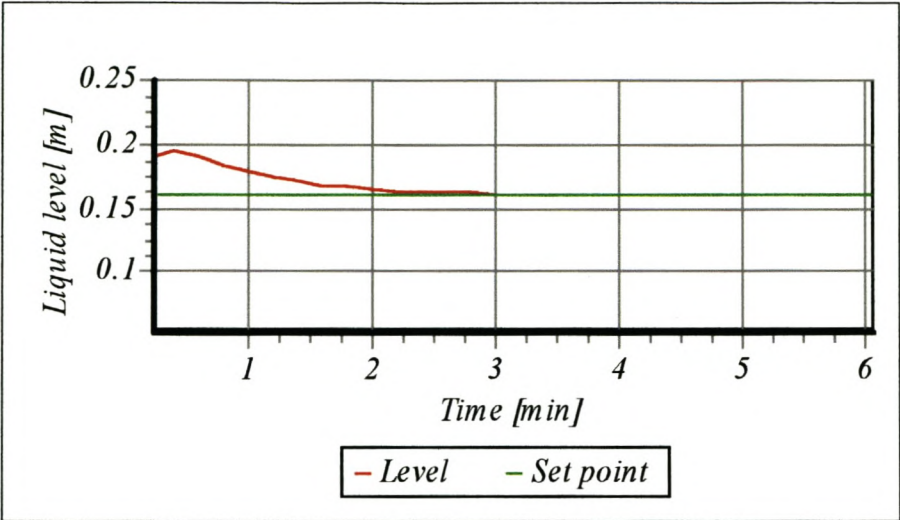


Figure 5-91 - Liquid level transient response from initial condition set 2 (section 6.6.1)

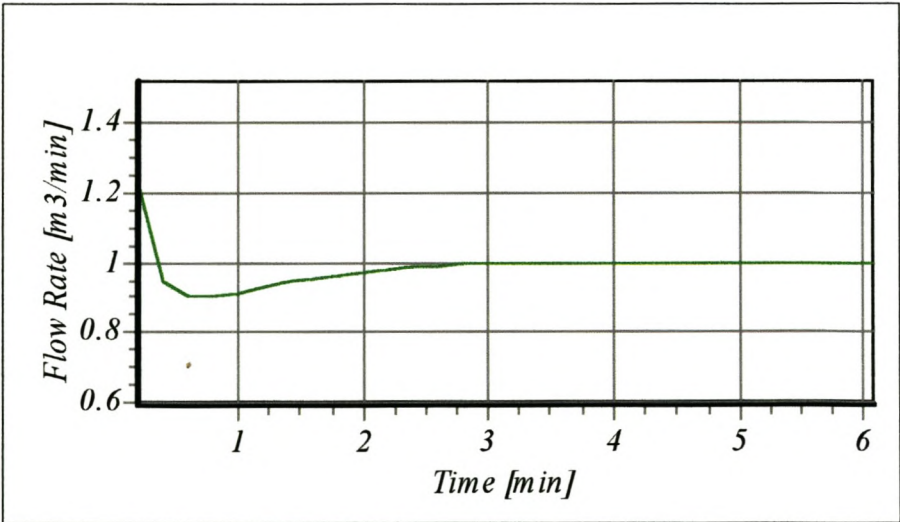


Figure 5-92 - Flow rate control action for initial condition set 2 (section 6.6.1)

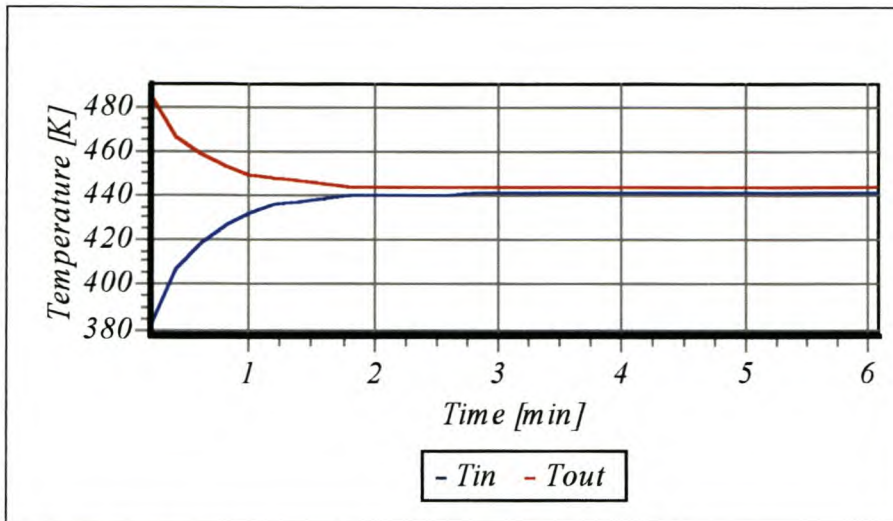


Figure 5-93 - Inlet temperature control action and reactor outlet temperature transient response. Initial condition set 2 (section 6.6.1)

5.4.6.3 Robustness analysis of the SANE direct method approach

Psichogios & Ungar (1991) (section 5.4.4) highlighted many difficulties in establishing neural *network* structures that performed satisfactorily in existing control structures. Evolving direct approach *neurocontrollers* with SANE posed none of the above difficulties. No consideration needed to be given to the invertability of the forward model - the state space was not partitioned to allow for unique inverse mappings to avoid inversion around a singular point. Rigorous on-line inversion of the forward model was also not required. No filters or classical control elements needed to be added to the control loop to ensure robust performance or eliminate *state*. Yet, superior performance is maintained from a response and a robustness viewpoint, even in the more complex *MIMO* control problem. SANE's focus on a more qualitative (implicit) rather than quantitative (explicit) model reference approach (section 5.1.2), could result in more robust *neurocontrollers* being generated than for the explicit direct inverse control approach.

In the explicit direct inverse control approach the training performance specification or constraints may be severe, leading to *convergence* problems and approximation errors. "Perfect control" occurs when $\frac{y_p}{r} = 1$ or $y_p = r$. "Perfect control" is, however, impossible as the controller would require an infinite gain. For the explicit model

reference control training structure, an unrealistic explicit response expectation (given the system dynamics) results when faster dynamics are required than may be realistically achieved. Faster dynamics require large controller gains, which could result in large *manipulated variable* control actions that may saturate the final control elements. Such large control actions could bring about instability should the inverse neural *network* be an imperfect representation of the true inverse. The appropriate choice of reference model in the explicit training structure (figure 4-2) is paramount, requiring prior knowledge of the system response to avoid iterative experimentation.

The implicit learning approach adopted by SANE in the *fitness* function, does not explicitly require the closed loop response to match an exact model reference. The obtained response is only implicitly related to the *ITAE* responses.

5.4.6.4 Model Parameter Perturbation analysis

During the training of the *neurocontroller* for the nonisothermal reactor each *neurocontroller* was presented with a conversion set point input. As the conversion set point of 0.508 is the optimum for the reactor system, the set point input was not varied during learning. The set point value is also implicitly included in the *fitness* function, making a constant set point input to the *neurocontroller* over an entire *evaluation* of little consequence in aiding the SANE algorithm to learn optimal behaviour in the *environment*. Hidden *neuron* connections may be connected with greater consequence to the task to be performed to other input or output *neurons* by the SANE algorithm. A constant set point as input over an entire *evaluation*, thus typically results in the particular input node being "disconnected" from the neural *network*. The set point input value is thus not used in the internal representation in such a manner as to allow for set point changes during operation. The constant input is effectively disregarded by the evolved *neurocontroller*.

A *neurocontroller* which regards the set point input as valuable information, is likely to utilise the deviation error from set point in the internal representation for the development of control behaviour. The deviation from set point may play a significant role in determining the controller output, as is the case for classical controllers. A disregard for the set point input to the neural *network*, however, allows for a different

mechanism to determine the behaviour of the *neurocontroller*. A different feature in the state space needs to be utilised to drive the process to set point.

The *neurocontroller* for the nonisothermal reactor was required to reach the optimum operating point. The *neurocontroller* would appear to have identified, during learning, the reactor outlet temperature as a significant contributor to reaching the optimum conversion. No set point was provided for the reactor outlet temperature. The neural *network* learned that controlling the reactor outlet temperature by manipulating the inlet temperature, indirectly determines the conversion in a significant fashion. The contribution of the reactor outlet temperature to good controller performance, is thus appreciable. This follows directly from the significant temperature dependence of the rate equations (equations 5-18 & 5-19). The neural *network* thus developed an internal representation during evolution, which drives the reactor outlet temperature to a set point region, that was "created" implicitly in the internal representation by SANE. This is evident in the below perturbation analysis as demonstrated in figure 5-94 to figure 5-95 and figure 5-96 to figure 5-97.

Figure 5-94 and figure 5-95 represent a perturbation of -10% in the parameter E1, while figure 5-96 and figure 5-97 represent a perturbation of +10% in the parameter E1. From figure 5-94 and figure 5-96 it is apparent that the conversion set point input node value played an insignificant role during learning in obtaining the optimum conversion of 0.508. Should the *neurocontroller* have developed an internal representation to regard the set point conversion input of 0.508, the large *state* would not be observed in figure 5-95. From figure 5-95 and figure 5-97 it is evident that the *neurocontroller* basis much of its behaviour on attaining a reactor outlet temperature in the region of 443 K (optimum reaction temperature for the nominal parameter case).

This evolved control *policy's* focus on the reaction outlet temperature results in the near maximum conversion being obtained for the particular perturbations, as illustrated in figure 5-74. This results as optimum conversion values for the considered perturbations lie in a narrow temperature range around 440 K.

A comparison between figure 5-80(a) and figure 5-96 reveals a similar response to the NLMPC1 with a prediction horizon of 15. From figure 5-97 the inlet temperature does not display the oscillatory behaviour as is evident in figure 5-80(b). This is indicative of the *neurocontroller's* ability to accurately predict the process gain sign change, despite the perturbation in the kinetic parameters of the model.

A comparison between figure 5-81(a) and figure 5-94 displays the control strategy difference between the SANE developed controller and the *MPC* implementation. In the *MPC* implementations the controller behaviour is coupled to the error from the set point value. The *MPC* controller attempts to maintain the conversion set point of 0.508 by reducing the reactor temperature and thereby operating at a lower conversion than is possible for the given perturbation in the system. The SANE controller's developed control *policy* focuses on controlling the reactor outlet temperature, resulting in operation that displays a large *state* from the input set point, but a steady state in close proximity to the optimum conversion for the specified perturbation model parameters.

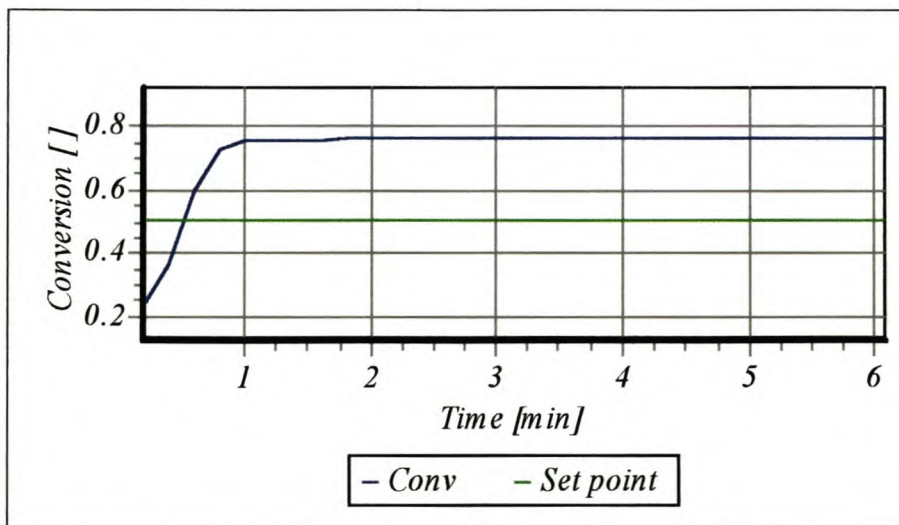


Figure 5-94 - Neurocontroller response to a process/model mismatch perturbation of -10% in E_1 .

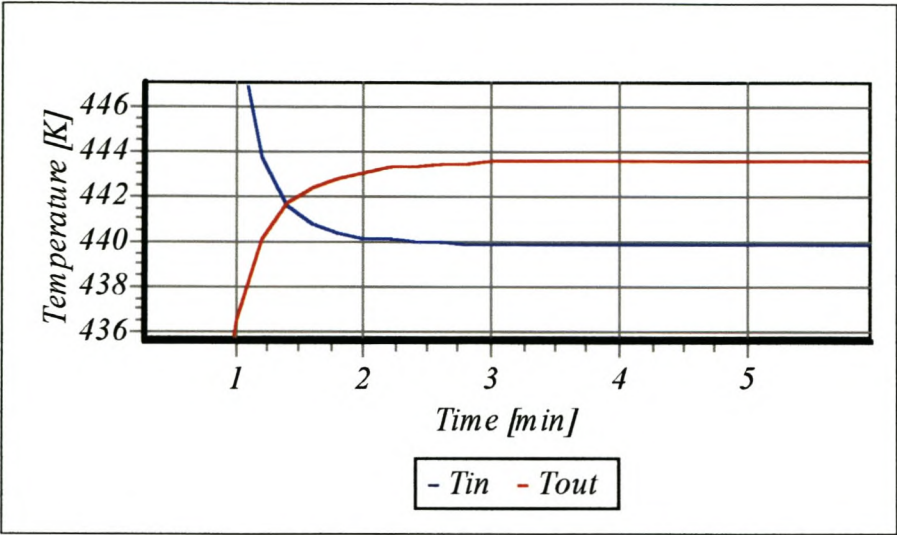


Figure 5-95 - Inlet temperature control action and reactor outlet temperature response for a process/model mismatch perturbation of -10% in E_1 .

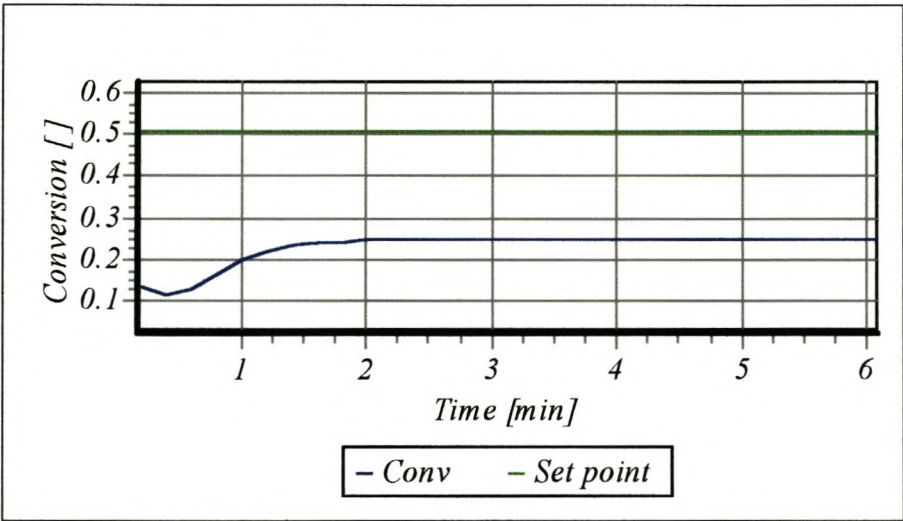


Figure 5-96 - Neurocontroller response to a process/model mismatch perturbation of +10% in E_1 .

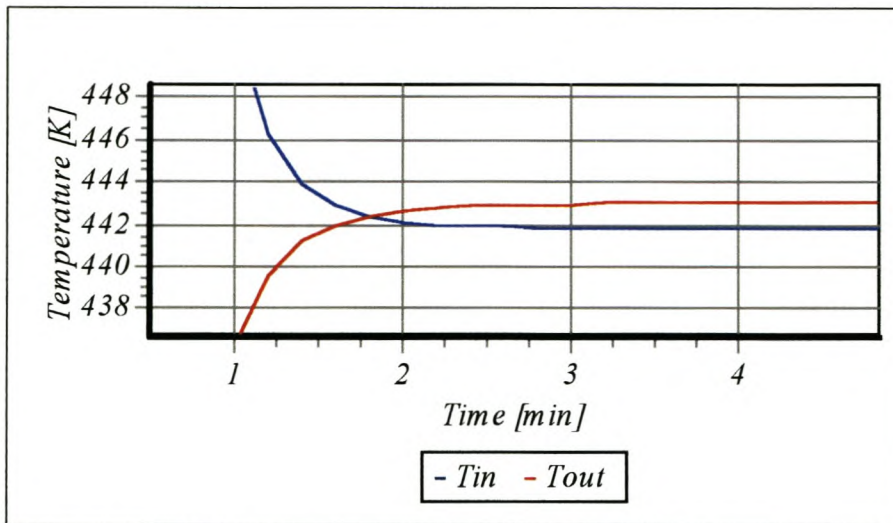


Figure 5-97 - Inlet temperature control action and reactor outlet temperature response for a process/model mismatch perturbation of -10% in E_2 .

5.5 VAN DE VUSSE REACTOR

5.5.1 Process Description

The control of CSTRs in which the reactions are governed by Van de Vusse type kinetics (section 1.3.1), have been extensively researched. In the following analysis, A is the educt, B the desired product, C and D are unwanted byproducts.



From a design perspective the objective is to make k_2 and k_3 small in comparison to k_1 by appropriate choice of catalyst and reaction conditions. The concentration of B in the product may be controlled by the manipulating the inlet flow rate and/or the reaction temperature.

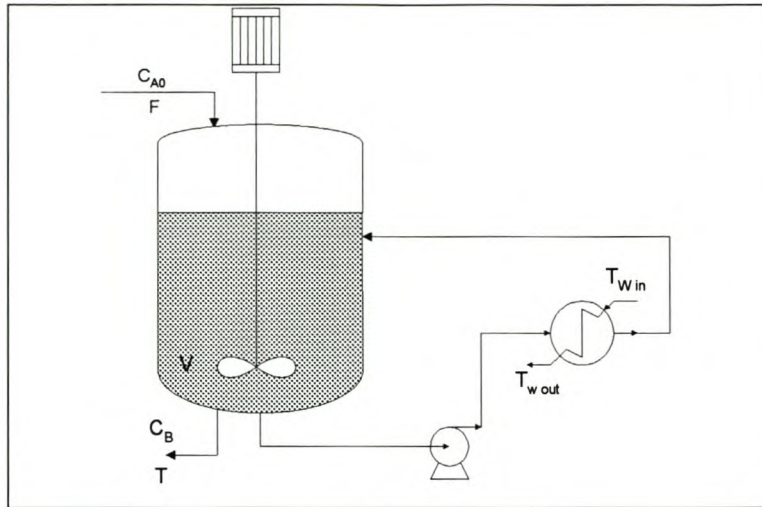
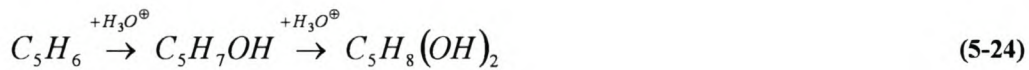


Figure 5-98 - Externally cooled CSTR with Van de Vusse reactions

Engel & Klatt (1993) considered the production of cyclopentanol as illustrated below.



The cyclopentenol is produced in a CSTR from cyclopentadiene(A) by acid-catalysed electrophilic addition of water in dilute solution. Due to the strong reactivity of the educt (A) and the product (B), dicyclopentadiene (D) is produced by the Diels-Alder reaction as a side product, and the cyclopentanediol (C) as a consecutive product by addition of another water molecule.

The educt flow contains only cyclopentadiene in low concentration, C_{A0} . Assuming constant density and an ideal residence time distribution within the reactor, the mass balance equations for the relevant concentrations of cyclopentadiene and of the desired product cyclopentanol, C_A and C_B , are as follows.

$$\frac{dC_A}{dt} = \frac{F}{V_R} \cdot (C_{A0} - C_A) - k_1 \cdot C_A - k_3 \cdot C_A^2 \quad (5-26)$$

$$\frac{dC_B}{dt} = -\frac{F}{V_R} \cdot C_B + k_1 \cdot C_A - k_2 \cdot C_B \quad (5-27)$$

Here V_R is the volume of the reactor, and F is the volumetric flow through the reactor. The energy balance yields the following differential equation for the temperature of the reactor content.

$$\begin{aligned} \frac{dT}{dt} = & -\frac{1}{(\rho \cdot C_p)} \left[k_1 \cdot C_A \cdot \Delta H_{R_{AB}} + k_2 \cdot C_B \cdot \Delta H_{R_{BC}} + k_3 \cdot C_A^2 \cdot \Delta H_{R_{AD}} \right] + \\ & \frac{F}{V_R} \cdot (T_{in} - T) + \left(\frac{k_w \cdot A_R}{\rho \cdot C_p \cdot V_R} \right) \cdot (T_w - T) \end{aligned} \quad (5-28)$$

where ΔH_{Rij} are the various reaction enthalpies (negative for exothermic reactions) and T_{in} is the temperature of the inlet stream to the reactor. The temperature, T_w , of the reactor coolant, from which a certain amount of heat, Q , is removed by an external heat exchanger is given by equation 5-29 below.

$$\frac{dT_w}{dt} = \frac{1}{(m_w \cdot C_{p,w})} \cdot [Q + k_w \cdot A_R \cdot (T - T_w)] \quad (5-29)$$

Here k_w represents the overall heat transfer coefficient, A_R the surface area of the cooling jacket and m_w is the mass of the coolant. The rate coefficients k_1 , k_2 and k_3 depend on the reaction temperature via Arrhenius' law and are given by the equations listed in table 5-6.

Table 5-6 - Model parameter formulations for the Van de Vusse reactions

<i>Description</i>	<i>Formulation</i>	<i>Unit</i>
Reaction enthalpy $A \rightarrow B$	$\Delta H_{R_{AB}} = (4.2 \pm 2.36)$	$\left[\frac{kJ}{mol}\right]$
Reaction enthalpy $B \rightarrow C$	$\Delta H_{R_{BC}} = -(11.0 \pm 1.92)$	$\left[\frac{kJ}{mol}\right]$
Reaction enthalpy $A \rightarrow D$	$\Delta H_{R_{AD}} = -(41.85 \pm 1.41)$	$\left[\frac{kJ}{mol}\right]$
Reaction rate constant $A \rightarrow B$	$k_1 = (1.287 \pm 0.04) \cdot 10^{12} \cdot e^{\left(\frac{-9.758.3}{T}\right)}$	$[h^{-1}]$
Reaction rate constant $B \rightarrow C$	$k_2 = (1.287 \pm 0.04) \cdot 10^{12} \cdot e^{\left(\frac{-9.758.3}{T}\right)}$	$[h^{-1}]$
Reaction rate constant $A \rightarrow D$	$k_3 = (9.043 \pm 0.27) \cdot 10^9 \cdot e^{\left(\frac{-8560}{T}\right)}$	$[h^{-1}]$
Reactor content density	$\rho = (0.9342 \pm 4.0 \cdot 10^{-4})$	$\left[\frac{kg}{l}\right]$
Reactor content heat capacity	$C_p = (3.01 \pm 0.04)$	$\left[\frac{kJ}{(kg \cdot K)}\right]$
Overall heat transfer coefficient	$k_w = (4032 \pm 120)$	$\left[\frac{kJ}{(h \cdot m^2 \cdot K)}\right]$
Heat exchanger area	$A_R = 0.215$	$[m^2]$
Reactor Volume	$V_R = 0.01$	$[m^3]$
Coolant mass	$m_w = 5$	$[kg]$
Coolant heat capacity	$C_{p_w} = (2.0 \pm 0.05)$	$\left[\frac{kJ}{(kg \cdot K)}\right]$

The product concentration C_B of the reactor described by the equations of state, is to be controlled with the following constraints on product purity and the *manipulated variables*, F and Q .

$$0.7 \leq C_B \leq 0.95 \left[\frac{mol}{l} \right] \quad (5-30)$$

$$50 \leq F \leq 350 \left[h^{-1} \right] \quad (5-31)$$

$$-8500 \leq Q \leq 0 \left[\frac{kJ}{h} \right] \quad (5-32)$$

The concentration of the educt in the inflow, C_{A0} , is considered as a disturbance and may vary in the range -

$$4.5 \leq C_{A0} \leq 5.7 \left[\frac{mol}{l} \right] \quad (5-33)$$

T and C_B are assumed to be available for measurement, but not C_A . The disturbance C_{A0} is also assumed unmeasured. The control of C_B is desired such that any value in the range specified for C_B , must be attained without steady state error for all values of C_{A0} (Engell & Klatt, 1993). The control law needs to be robust with respect to the uncertainties of the model parameters. Changes to the set point of C_B should lead to well damped smooth transient responses which reach the new steady-state value as fast as possible, at most after 15 minutes (Engell & Klatt, 1993).

Around the chosen steady state values, significant nonlinear behaviour is apparent in both a dynamic and steady-state sense. For the outlet concentration C_B a significant change in steady state gain is observed for small changes in set point. For the reactor temperature, T , a large variation in the gain is observed in the vicinity of the setpoint. Dynamically, the system exhibits nonminimum phase behaviour. The characteristic inverse response of nonminimum phase systems (as a result of the RHP zero) is observed in the open-loop response (Harris & Palazoglu, 1998).

5.5.2 Control Policies cited for the nonminimum phase reactor

Engell & Klatt (1993) considered the case where both the flow through the reactor and the amount of heat removed are available as *manipulated variables* to control the product specification in the specified range. To eliminate the temperature dependence of the reaction rates, Engell & Klatt (1993) introduced a fast inner temperature control loop. This controller keeps the reactor temperature approximately constant by manipulating the output of the external cooling unit. Maintaining a constant reactor

temperature reduces the dynamics of the concentration of the product to a second order nonlinear system (*SISO*), which makes controller development for the product concentration control, less difficult. The *nonminimum phase* behaviour is, however, still present in the reduced system. A gain scheduling controller was developed to accomplish the control task. The developed controller needed to be rigorously analysed for stability, as gain scheduling control schemes may become unstable under certain conditions.

Harris & Palazoglu (1998) considered the *MIMO* control system in which the set points are the reactor temperature and the reactor concentration of B. The *manipulated variables* are both the external heat exchanger duty Q_w and the reactor flow rate F . The concentration of C_A in the feed and in the reactor was considered unmeasured.

Harris et al. (1998) consider a nonlinear controller design based upon functional expansion models. Two different decoupling schemes were considered, ideal and one-way decoupling. Ideal decoupling allows for no interaction between the controllers. The one-way scheme decoupled the outlet concentration from the reactor temperature. In the one-way arrangement, when set point changes are made in the outlet concentration, deviations in the reactor temperature are not allowed. The nonminimum phase behaviour is primarily observable in the concentration of B. The one-way decoupling consequently isolates the effect of this inverse behaviour in C_B 's response. Both decoupling schemes will only be approximate, due to plant/model mismatches resulting from FEx model approximation.

5.5.3 Concluding remarks for control strategies considered in previous research

The Van de Vusse reactor setup as illustrated in figure 5-98, represent a *MIMO* control problem comprising two integrated unit operations. The process model presented by Engell & Klatt (1993) is detailed in that an expected operating range for each model parameter is provided, allowing for Monte-Carlo type robustness analyses of *neurocontroller* performance. The considered control strategies (section 5.5.2)

incorporate a design constraints on the reactor temperature (to remove the nonlinearity introduced by the Arrhenius equation) to facilitate controller development on a simplified model. Decoupling schemes also needed to be considered to develop an effective control *policy*.

5.5.4 Symbiotic, Adaptive Neuro Evolution (SANE)

5.5.4.1 Neurocontroller evaluation learning

A *neurocontroller* was developed with 3 input nodes, 8 hidden nodes and 2 output nodes. The three input nodes represent the concentration of B in the reactor, the desired set point for C_B and the reactor temperature. The two output nodes specify the two *manipulated variables*, the reactor flow rate and the cooling duty provided by the external heat exchanger. The initial plant state is randomly generated in a limited region of the state space, which may represent a process condition not realistically obtainable in practice. Control from such initial conditions is necessarily more complex, as the initial state variable may have large unrealistic ratios. A typical learning trial response for the best developed *neurocontroller* is illustrated in figure 5-99 to figure 5-103.

The *nonminimum phase* behaviour in C_B is effectively rejected by the *neurocontroller* (figure 5-100). No inverse response is evident during set point changes.

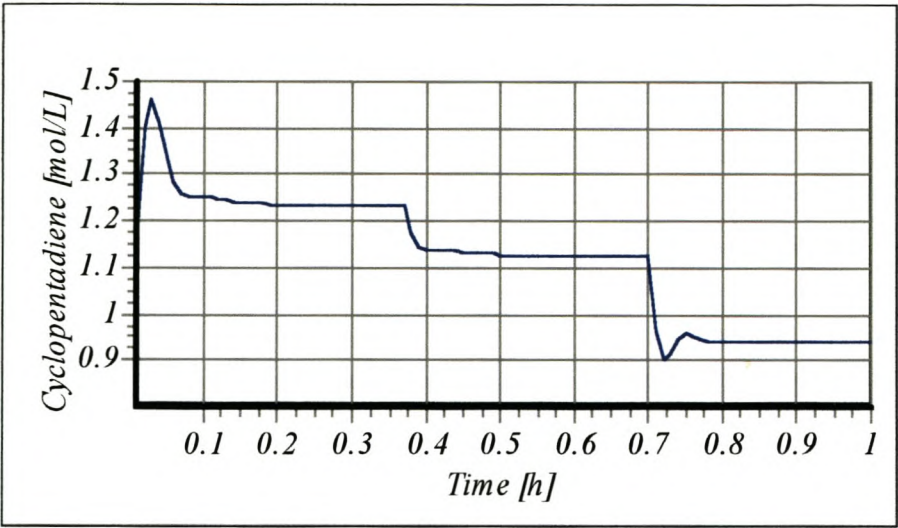


Figure 5-99 - Educt, A, transient response from an initial condition $C_A = 1.14 \text{ mol/L}$, $C_B = 0.82 \text{ mol/L}$, $T_0 = 410.6 \text{ K}$ & $T_w = 375.74 \text{ K}$.

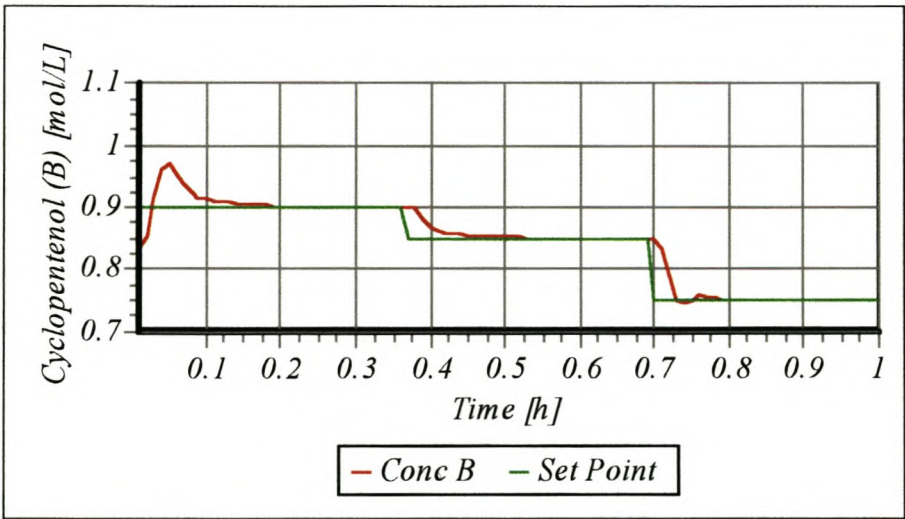


Figure 5-100 - Product, B, transient response from an initial condition $C_A = 1.14 \text{ mol/L}$, $C_B = 0.82 \text{ mol/L}$, $T_0 = 410.6 \text{ K}$ & $T_w = 375.74 \text{ K}$.

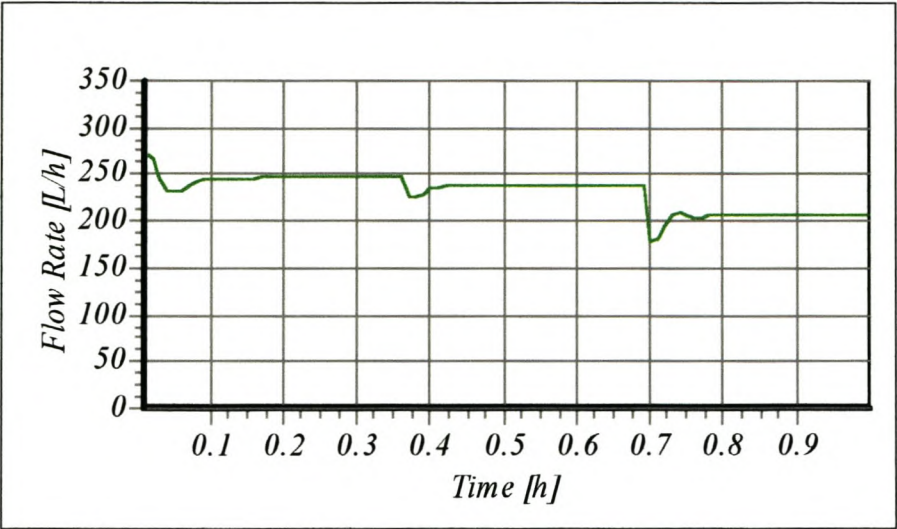


Figure 5-101 - Inlet flow rate control action for initial condition $C_A = 1.14$ mol/L, $C_B = 0.82$ mol/L, $T_0 = 410.6$ K & $T_w = 375.74$ K.

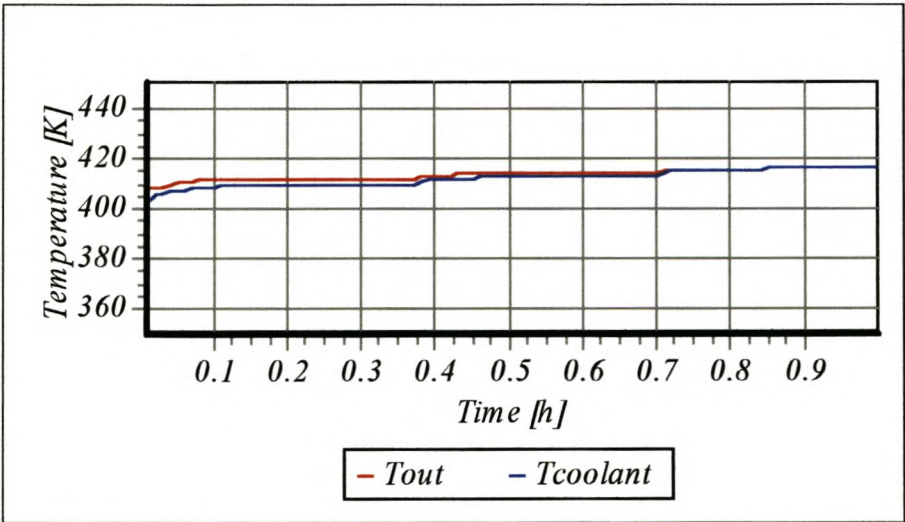


Figure 5-102 - Reactor system temperature transient responses from initial condition $C_A = 1.14$ mol/L, $C_B = 0.82$ mol/L, $T_0 = 410.6$ K & $T_w = 375.74$ K.

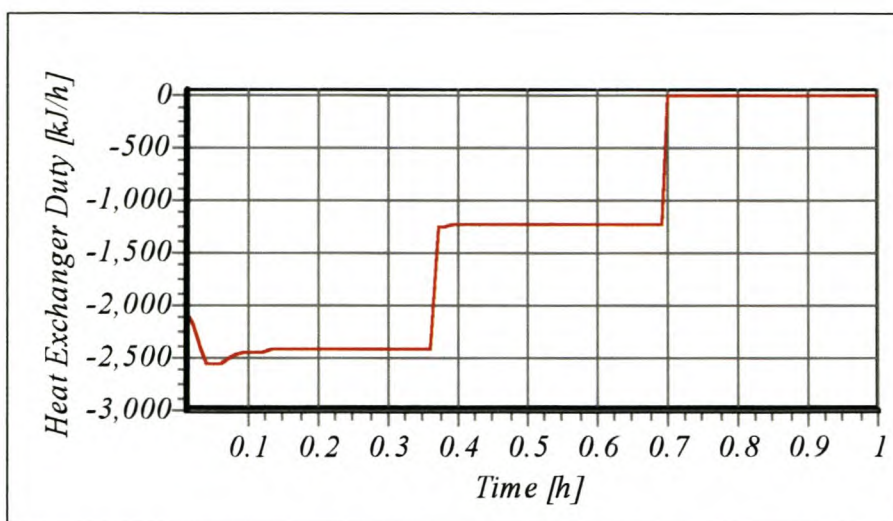


Figure 5-103 - External heat exchanger heat duty control action for initial condition $C_A = 1.14$ mol/L, $C_B = 0.82$ mol/L, $T_0 = 410.6$ K & $T_w = 375.74$ K.

5.5.4.2 Servo response characteristics

In accordance with the stated control objective of reaching any desired C_B set point without *state* (section 5.5.1), the *neurocontroller's* performance was gauged at obtaining three unlearned steady states. Two of these steady states ($C_B = 0.7 \frac{\text{mol}}{\text{L}}$ & $C_B = 0.95 \frac{\text{mol}}{\text{L}}$) represent an extrapolation outside the learned operating range. The transient responses for these set point changes are illustrated in figure 5-104 to figure 5-108.

The extrapolation set point change to $C_B = 0.95 \frac{\text{mol}}{\text{L}}$ displayed the poorest performance with a *state* of $0.02 \frac{\text{mol}}{\text{L}}$ in C_B . This may be attributed to unforeseen process steady state gain changes in this region of the state space, which resulted in poor learning extrapolation. In contrast, the other extrapolation set point change to $C_B = 0.7 \frac{\text{mol}}{\text{L}}$, only represented an offset of $0.002 \frac{\text{mol}}{\text{L}}$ in C_B . This may be attributed to a more predictable process steady state gain changes in this region of the state space - in this case the *neurocontroller* is able to extrapolate effectively. The set point change

to $C_B = 0.8 \frac{\text{mol}}{\text{L}}$ (which represents learning interpolation) resulted in a *state* of $0.005 \frac{\text{mol}}{\text{L}}$.

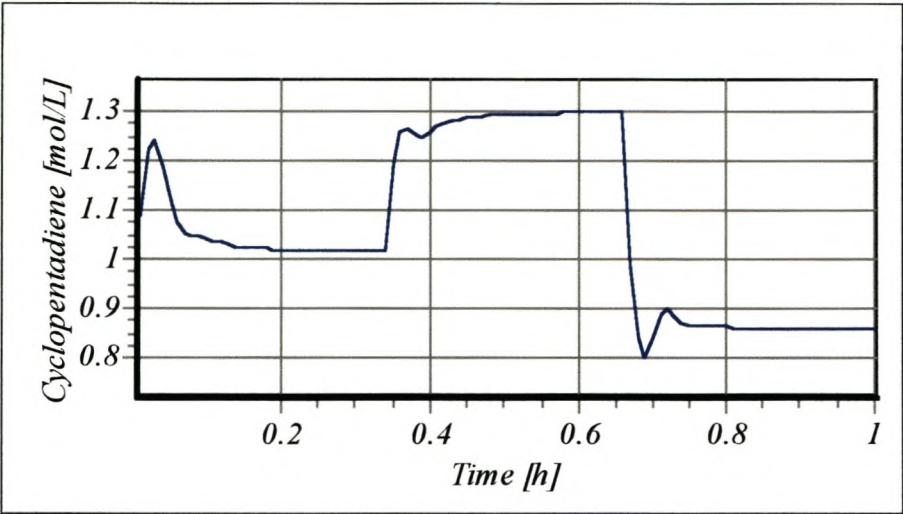


Figure 5-104 - Educt, A, concentration transient response for three unlearned set point changes $SP_1 = 0.8$, $SP_2 = 0.95$ & $SP_3 = 0.7$.

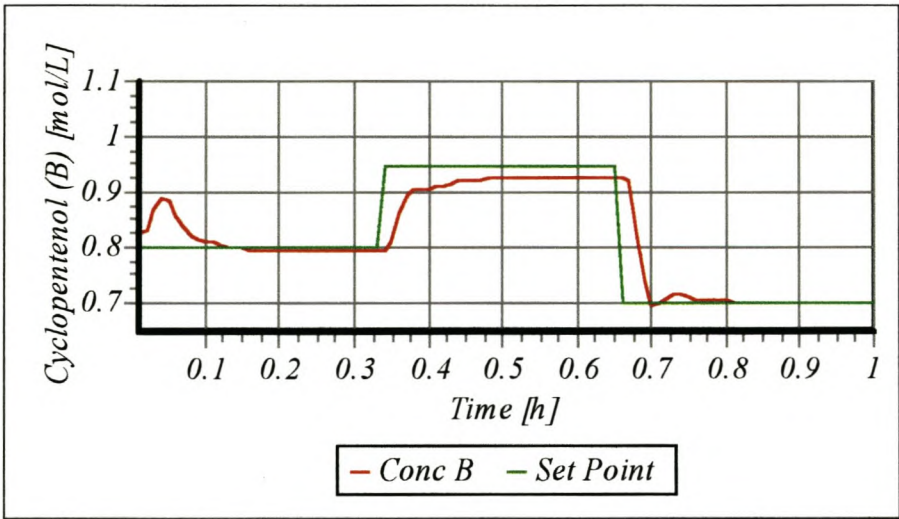


Figure 5-105 - Product, B, concentration transient response for three unlearned set point changes $SP_1 = 0.8$, $SP_2 = 0.95$ & $SP_3 = 0.7$.

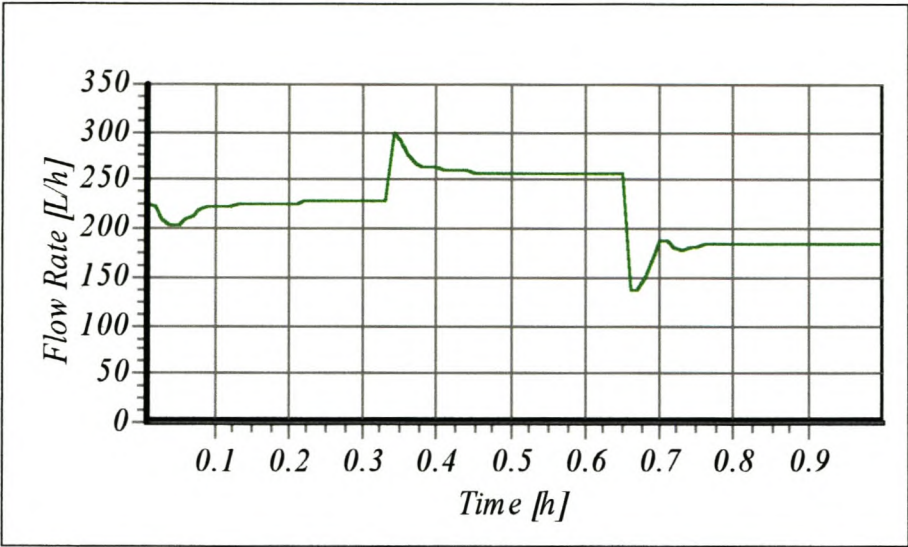


Figure 5-106 - Inlet flow rate control action for three unlearned set point changes $SP_1 = 0.8$, $SP_2 = 0.95$ & $SP_3 = 0.7$.

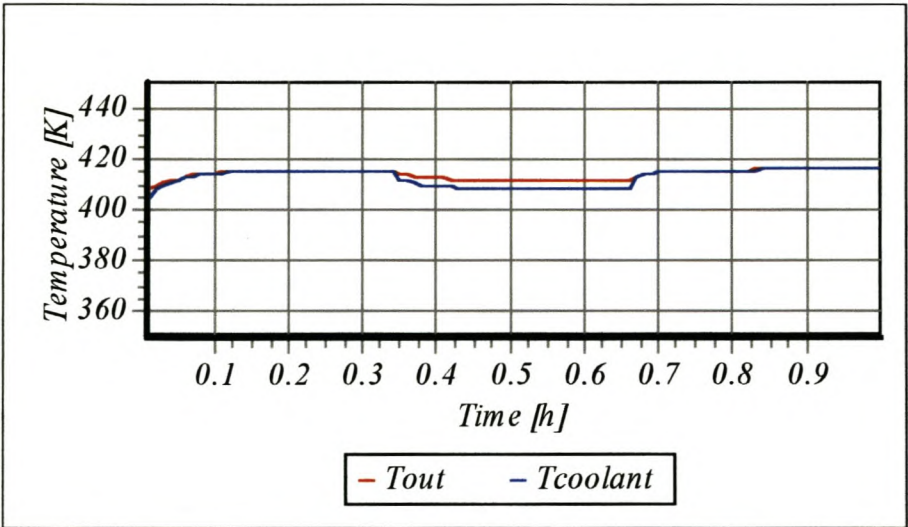


Figure 5-107 - Temperature transient responses for three unlearned set point changes $SP_1 = 0.8$, $SP_2 = 0.95$ & $SP_3 = 0.7$.

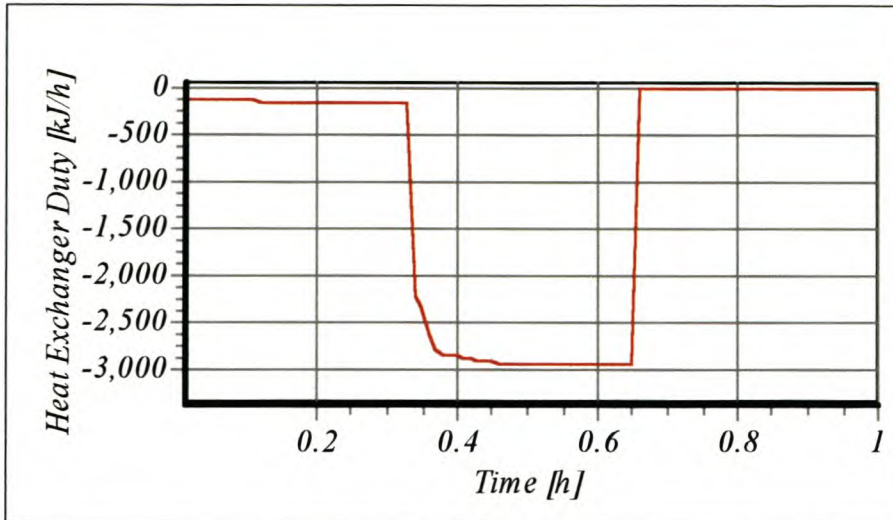


Figure 5-108 - Heat duty control action for three unlearned set point changes $SP_1 = 0.8$, $SP_2 = 0.95$ & $SP_3 = 0.7$.

5.5.4.3 Feed concentration disturbance rejection

As required by the control objective presented in section 5.5.1, the developed controller needs to be robust over a wide range of inlet feed concentrations. The inlet feed concentration was allowed to vary in a *gaussian* fashion around the mean $5.1 \left[\frac{\text{mol}}{\text{dm}^3} \right]$, with a standard deviation of $0.25 \left[\frac{\text{mol}}{\text{dm}^3} \right]$ (figure 5-109). The robust transient responses are illustrated in figure 5-110 to figure 5-114. The high frequency (at the sampling rate frequency) and magnitude of the feed disturbance is a worse case scenario to the reactor, as the transients for the reactor require longer transient decay times than provided for by the high frequency disturbance.

Of significance is the contribution of each *manipulated variable* to rejecting the feed disturbance. From figure 5-112 and figure 5-114 the *neurocontroller* would seem to regard the inlet flow rate as the most appropriate contributor to rejecting feed disturbances. As the CSTR is assumed perfectly mixed, an inlet disturbance (flow rate or concentration) has an immediate effect on reactor performance - no process lag time (dynamics) is allowed for in the model to allow for mixing effects. The dynamics for the external heat exchanger are, however, included in the model allowing for a delayed action-response. The *neurocontroller* has thus learned that faster corrective action in the modelled *environment* is possible by allowing for greater regulation

utilising the inlet flow rate to the reactor. This may not be the case in practice, as complex dynamics represent the mixing phenomenon in the reactor. Model limitations thus need to be considered in developing control strategies.

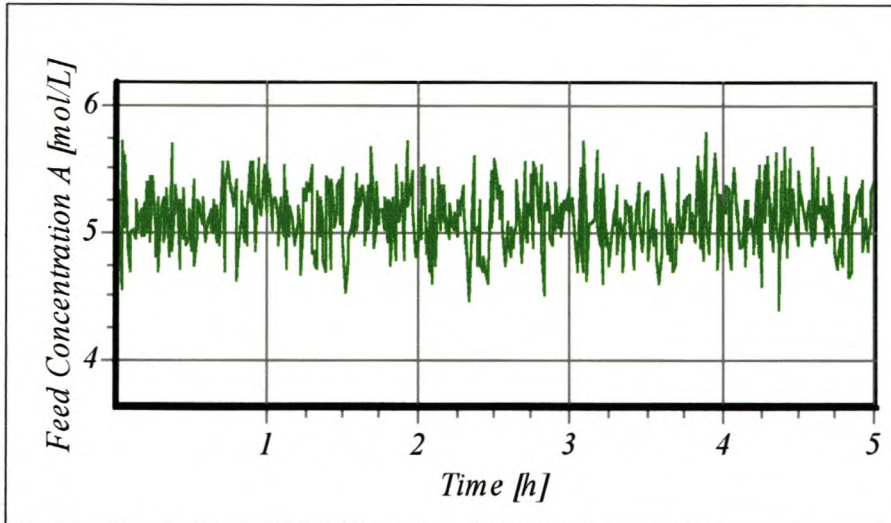


Figure 5-109 - Inlet feed concentration C_{A0} disturbance to the Van de Vusse reactor, with gaussian distribution around the mean 5.1, with a standard deviation of 0.25.

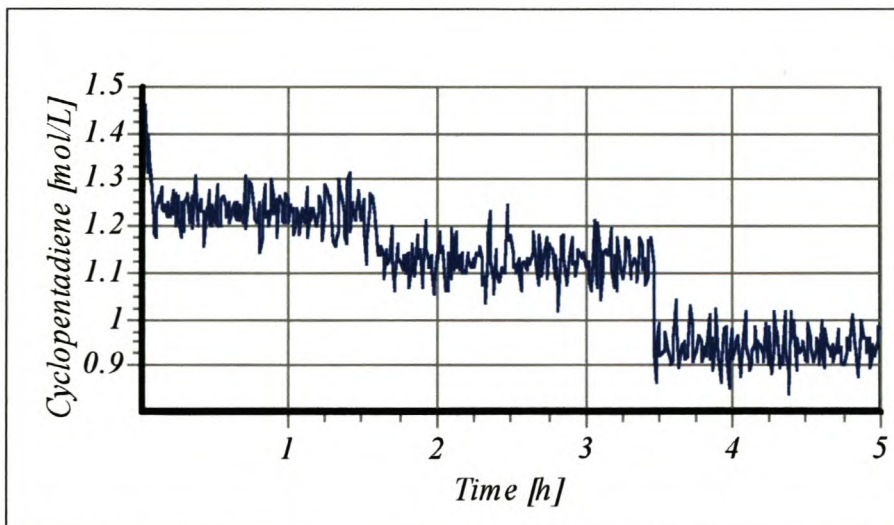


Figure 5-110 - Educt, A, concentration transient response with inlet feed concentration disturbances

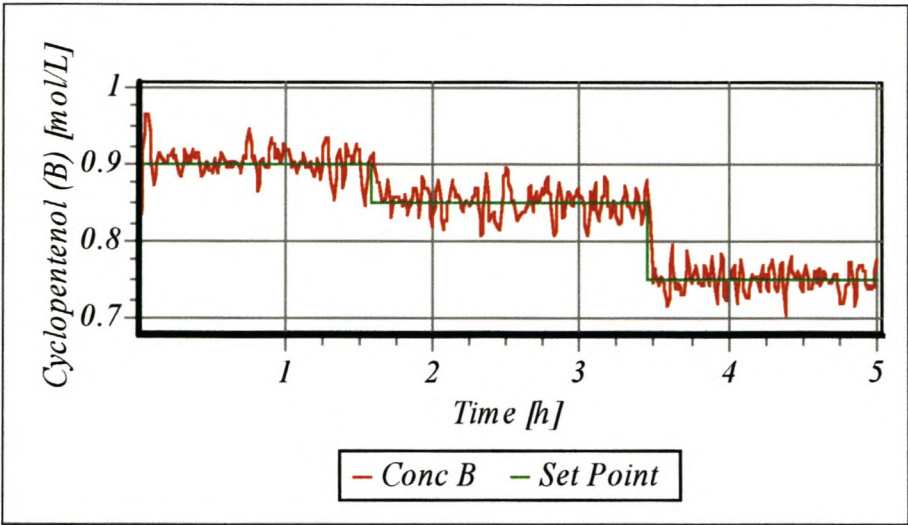


Figure 5-111 - Product, B, concentration transient response with inlet feed concentration disturbances

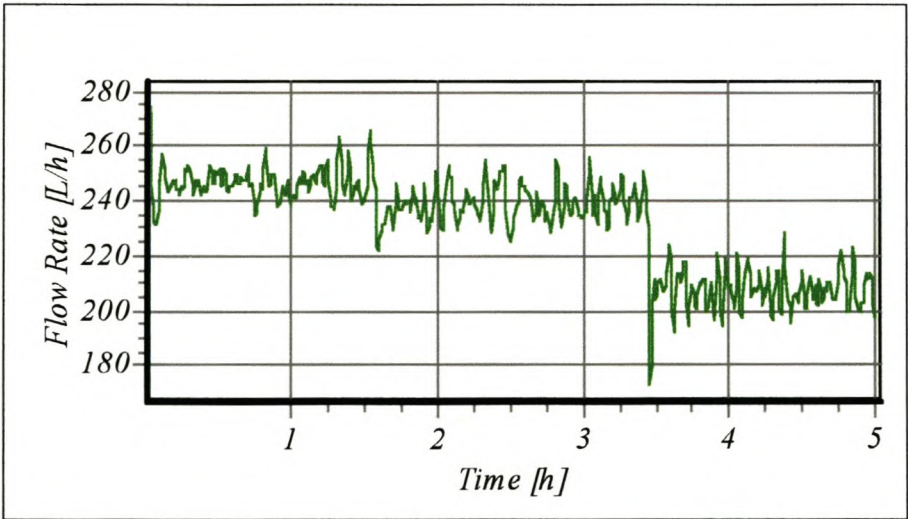


Figure 5-112 - Reactor inlet flow rate control action to inlet feed concentration disturbances.

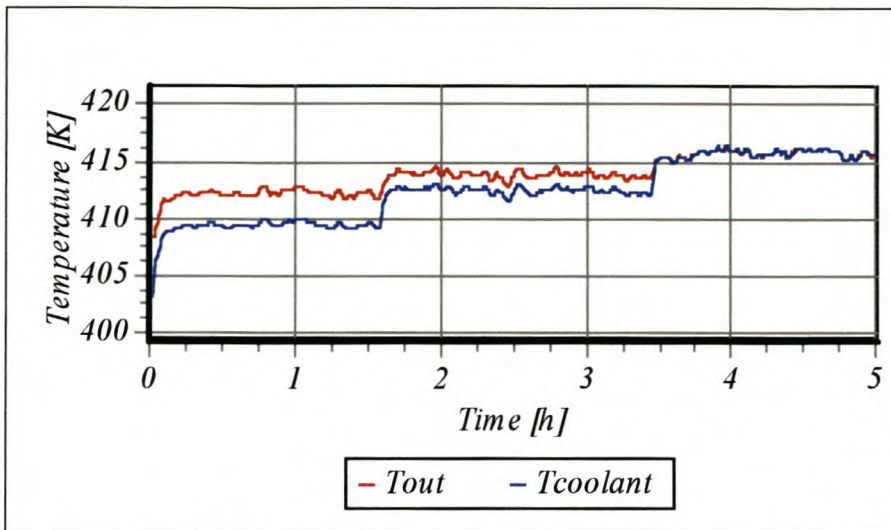


Figure 5-113 - Temperature responses for the reactor system with inlet feed concentration disturbances.

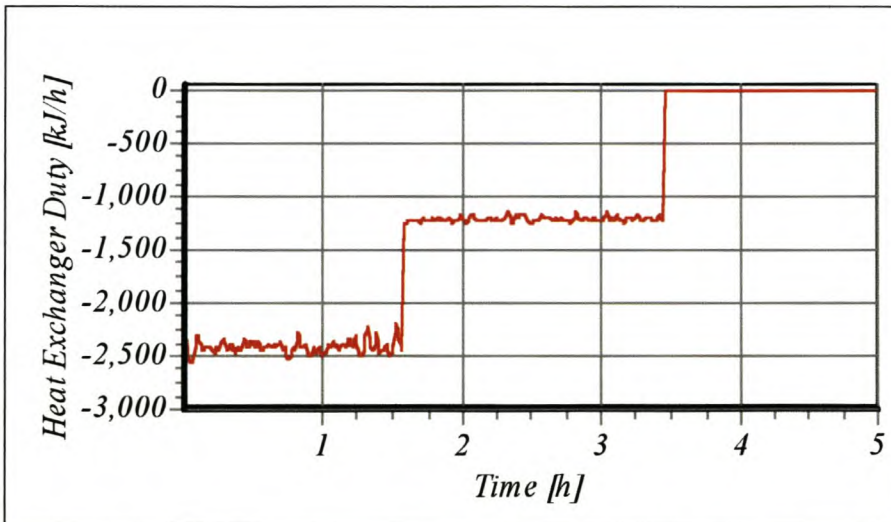


Figure 5-114 - Heat duty control action to inlet concentration feed disturbances

5.5.4.4 Disturbance rejection in multiple simultaneously varying model parameters

The Van de Vusse reactor model proposed by Engell & Klatt (1993) includes the expected variation ranges for each model parameter. In real world applications, physical properties and kinetic constants are unlikely to remain constant, due to changes in the state variables.

Robust controller design requires that the controller be robust for all possible encountered variable model parameters. Each model parameter in table 5-6 with a

variable range, was simultaneously allowed to assume a random *gaussian* distribution around the mean with 3σ representing the range limits, at each sample interval. This thus conforms to a mini, local Monte-Carlo robustness simulation over a single simulation trial. The response to this mini Monte-Carlo analysis is illustrated in figure 5-115 to figure 5-119. As illustrated in the transient responses of the state variables, random model parameter variations suggested by Engell & Klatt (1993), are effectively rejected at each sampling interval. The *neurocontroller* is thus expected to be robust over the full range of model parameter variation.

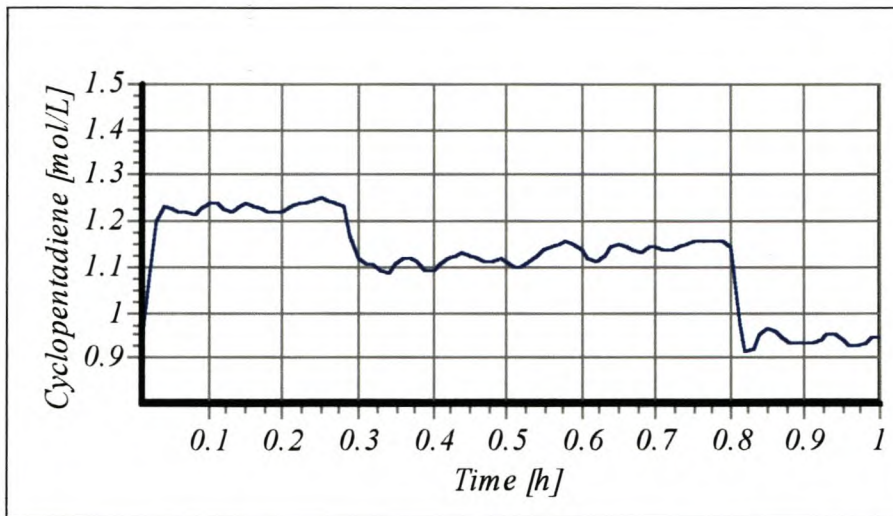


Figure 5-115 - Educt, A, transient response to multiple model parameter gaussian variance.

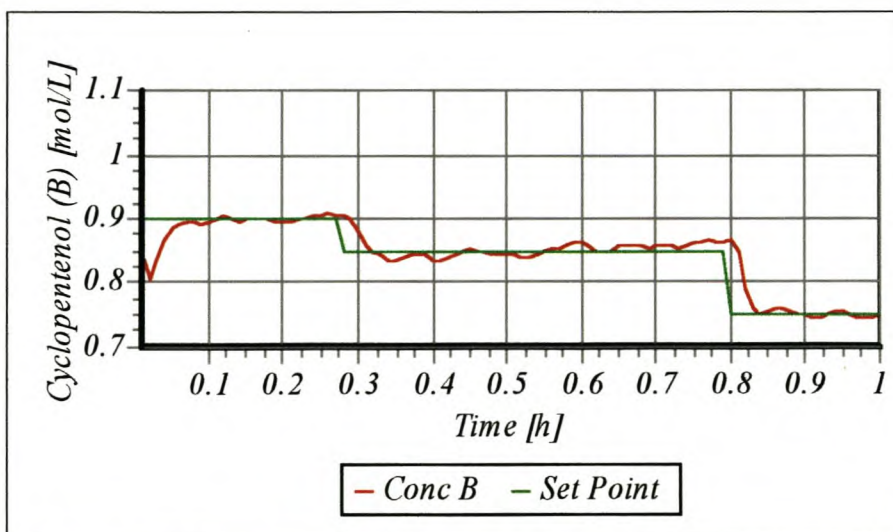


Figure 5-116 - Product, B, transient response to multiple model parameter gaussian variance

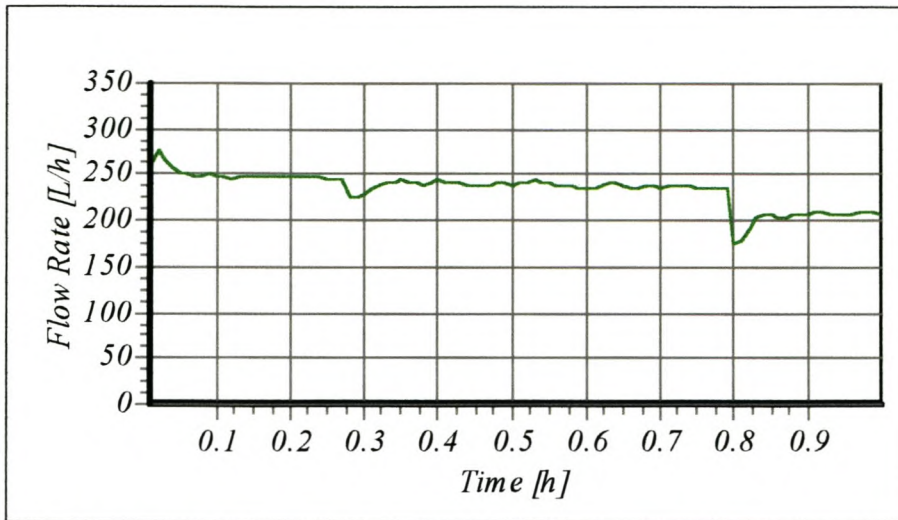


Figure 5-117 - Inlet flow rate control action response to multiple model parameter gaussian variance

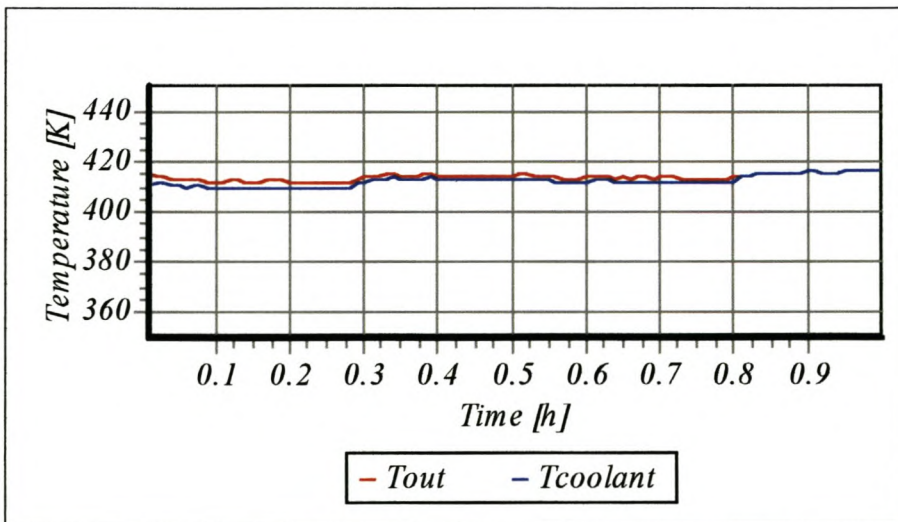


Figure 5-118 - Temperature response for the reactor system to multiple model parameter gaussian variance

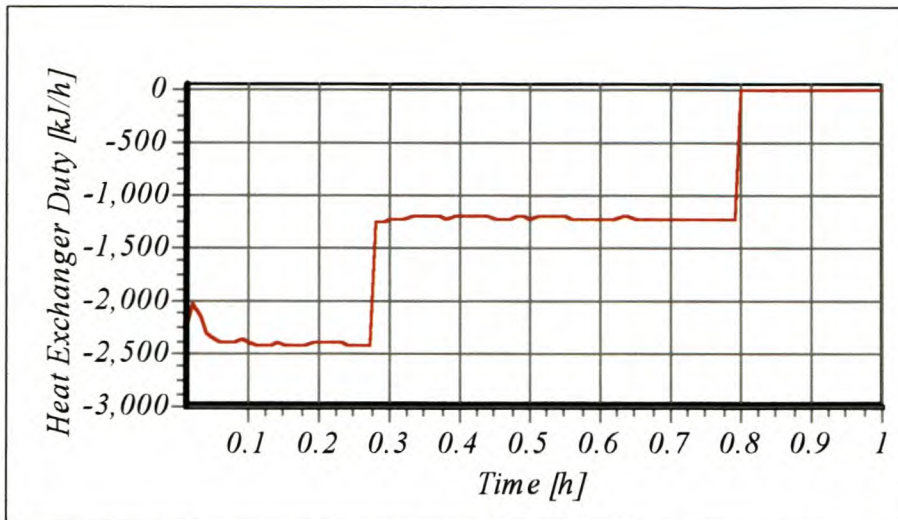


Figure 5-119 - Heat duty control action to multiple model parameter gaussian variance.

5.5.4.5 Increased economic benefit through unconstrained controller development

In the control implementations presented (section 5.5.2), a premise exists to maintain the reactor outlet temperature at a specified set point. Engell & Klatt (1993) incorporated the use of a fast inner temperature control loop to eliminate the effect of the reactor temperature on the rate equations. Harris & Palazoglu (1998) proposed a one-way decoupling scheme in which change in C_B do not result in change to the reaction temperature. This approach assumes a prior analysis of the model equations, to determine the optimum reaction temperature. The approach also effectively removes the reactor temperature as a degree of freedom in the system. In the *SANE* implementation the removal of degrees of freedom is not required to simplify the controller design methodology. No set point was thus provided for the outlet temperature to the *neurocontroller* as input. *SANE* was thus allowed to vary the reaction temperature as an additional degree of freedom, in response to disturbances or set point changes. This may be accomplished directly by the removal of heat duty, Q , from the reactor via an external heat exchanger or indirectly by manipulating the flow rate to the reactor. The cost (capital and operating) of incorporating an external heat exchanger into the process design was not considered in this analysis. A high cost of capital or high utility costs would impact the *SANE* controller design (if specified in the *fitness* function), to place less emphasis on the direct control, with more emphasis on indirect control, of reaction temperature. In this analysis the cost in

utilising direct and indirect control methods is considered equivalent in economic terms. The need for decoupling (if any) was determined implicitly by *SANE* during learning. No decoupling arrangement (constraint) needed to be specified explicitly. As the appropriate choice of decoupling arrangement often proves complex, this implicit determination of any required decoupling (while learning) is an advantage provided by the *SANE* approach.

Harris et al. (1998) considered the nominal set point of $C_B = 0.9 \frac{\text{mol}}{L}$. The prescribed process constraints $T = 407.3 \text{ K}$, resulted in a steady state operating point with a heat duty removal of $Q = -4496 \frac{\text{kJ}}{h}$ and $F = 188.3 \frac{L}{h}$. The control policy developed by *SANE* for the nominal set point as considered by Harris et al. (1998), resulted in a steady state operating point with a duty removal of $Q = -2405 \frac{\text{kJ}}{h}$ (figure 5-103) and $F = 247.45 \frac{L}{h}$ (figure 5-101) at an operating temperature of $T = 412.14 \text{ K}$ (figure 5-102). Not allowing the outlet reactor temperature to be constrained, thus has a significant economic impact on the operation of the reactor. The required heat duty removal is 46.5% less than the nominal heat duty at steady state, which implies lower capital expenditure (small heat exchanger) and reduced operating costs (utility costs). The higher flow rate with the same outlet concentration in B, thus results in a 31.4% higher production rate which implies increased revenue. The outlet concentration of A was slightly lower at $C_A = 1.232 [\frac{\text{mol}}{L}]$, than for the steady state concentration of $C_A = 1.235 [\frac{\text{mol}}{L}]$ obtained by Harris et al. (1998). A small fraction of 0.24% of educt, A, was thus additionally converted by the *SANE neurocontroller* to unwanted byproducts, which is considered negligible. The opportunity for the integration of process design and optimal process control is thus evident in the *SANE* approach.

5.6 CONCLUDING REMARKS FOR CHAPTER 5

The process control simulation studies served to verify the applicability of *SANE* as a means of developing optimal nonlinear controllers for highly nonlinear process systems. Each simulation study demonstrated *SANE*'s ability to generalise and remain robust over a wide range of operating conditions and in the presence of significant process uncertainty. The developed *neurocontrollers* could effectively reject a wide variety of process disturbance, such as feed disturbances and sensor noise. In comparison with established advanced control techniques, such as model predictive control, the developed *neurocontrollers* proved less sensitive to discrepancies between model and actual plant parameters. *SANE* was also utilised to integrate the optimal process design and process control development functions (conventionally two very different functions) into a single calculation step, which may result in significant economic benefit.

5.7 SYMBOLS FOR CHAPTER 5

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
a	Yield coefficient parameter	-
A_R	Heat transfer area	$[m^2]$
b	Yield coefficient parameter	-
C	Concentration	$[\frac{mol}{dm^3}]$
C_1	Dimensionless cell mass	-
C_2	Dimensionless substrate conversion	-
C_p	Specific heat capacity	$[\frac{kJ}{(kg \cdot K)}]$
C_{pw}	Coolant heat capacity	$[\frac{kJ}{(kg \cdot K)}]$
Da	Damkohler number	-
e	Error from set point	-
E_1	Forward reaction activation energy	$[K]$
E_2	Reverse reaction activation energy	$[K]$
F	Inlet flow rate to reactor	$[\frac{L}{h}]$
F_O	Outlet flow rate from reactor	-
h	Liquid level in reactor	$[dm]$
J	ITAE criteria minimisation function	-
k	Kinetic reaction constant	-
k_{01}	Forward reaction Arrhenius constant	$[min^{-1}]$
k_{02}	Reverse reaction Arrhenius constant	$[min^{-1}]$
k_w	Overall heat transfer coefficient	$[\frac{kJ}{(h \cdot m^2 \cdot K)}]$
K_1	Forward reaction kinetic rate constant	$[min^{-1}]$
K_2	Reverse reaction kinetic rate constant	$[min^{-1}]$
m_w	Coolant mass	$[kg]$
P	Pressure	-
Q	Cooling duty	$[\frac{kJ}{h}]$

r	Reaction rate	-
S	Substrate concentration	-
t	Time	[s]
T	Temperature	[K]
T	Dimensionless sampling interval	-
u ₁	Inlet reactor flow rate	$[\frac{1}{\text{min}}]$
u ₂	Feed inlet temperature	[K]
V	Reactor volume	[dm ³]
w	Inverse of Damkohler number	-
w ₁	Inlet flow rate for concentrated stream	$[\frac{dm}{s}]$
w ₂	Inlet flow rate for dilute stream	$[\frac{dm}{s}]$
x ₁	Conversion of reactant A	-
x ₂	Reactor outlet temperature	[K]
x ₃	Reactor level	[m]
X	Cell mass	-
X ₀	Inlet reactant overall concentration	$[\frac{mol}{dm^3}]$
y ₁	Reactor level set point	[m]
Y	Yield coefficient	-

Greek symbols

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
β	Weighting coefficient for bioreactor	-
γ	Weighting coefficient for bioreactor	-
Γ	Diagonal weight vector	-
μ	Specific growth rate	-
ρ	Reactor content density	$[\frac{kg}{l}]$
σ	Specific substrate consumption rate	-
τ	Dimensionless time	-

Superscript

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
SP	set point	-

Subscript

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
b	Reactant B	-
b1	Concentrated Feed B	-
b2	Dilute Feed B	-
c	Critical	-
F	Feed	-
S	Set point	-

6 MULTI-EFFECT BATCH DISTILLATION – REAL WORLD APPLICATION

OBJECTIVES OF CHAPTER 6

- Investigate the learning ability of *SANE* on a real world novel batch distillation column.
 - Compare a PI temperature control strategy with the developed *neurocontroller* control *policy*.
-

Batch distillation is one of the oldest unit operations in the chemical industry for separating liquid mixtures. Although it has been used for centuries in the production of alcoholic beverages, essential oils, perfume, pharmaceutical and petroleum products, the continuous process became the standard in the 1940's as economic pressure dictated the use of economies of scale. The renewed emphasis on the production of small amounts of product with high added value, has resulted in an increased use of batch distillation for specialised production. Batch distillation is frequently employed as a means of separating small amounts of high added value products in fine chemicals and pharmaceutical industries. Attractive features for the use of batch distillation are the simplicity of operation, flexibility and lower capital cost (Mujtaba, 1992).

From the viewpoint of energy consumption, batch distillation is generally inefficient compared with continuous distillation. A need arises for greater energy efficiency in batch distillation column designs. As indicated in section 1.3.3, for continuous distillation columns, considerable improvements are possible by utilising novel column configurations for separation services. The utilisation of these novel column configurations may reduce both the capital and operating costs associated with separation. For batch distillation systems unconventional column configurations have been proposed, which promise to deliver similar energy efficiency as for conventional columns. The complex and Multi-Effect Batch Distillation Column (*MEBAD*) represent such novel batch column configurations. With the advent of novel energy efficient batch distillation configurations, conventional rectification batch columns may, in many cases, be regarded as a case of overdesign as discussed in section 1.2.

Novel batch distillation column arrangements have, however, not been accepted into industrial practice. Design and operation uncertainties and ill understood dynamics present obstacles to industrial acceptance, as more complex control strategies are often required in such column implementation.

Section 6.1 describes two novel column configurations. Section 6.2 gives consideration to possible process applications for such novel column configurations. Section 6.3 discusses the economic potential of novel column configurations as compared to conventional application. Section 6.4 describes control policies proposed in academic literature. Section 6.5 details the experimental set-up and procedure followed in this study. Section 6.6 presents the results obtained from experimental data.

6.1 PROCESS DESCRIPTION

Two direct heat integrated novel batch distillation configurations are presented. Section 6.1.1 details the configuration of a complex column and section 6.1.2 describes the *generalisation* to the Multi-Effect Batch Distillation (*MEBAD*) configuration.

6.1.1 Batch Distillation in Continuous Distillation Columns

Davydyan et al. (1991) considered the case of batch fractionation in a two-section column with a central reservoir as illustrated in figure 6-1, where distillation of the mixture (charged to the reservoir) is performed with simultaneous withdrawal of distillate from the upper section and bottoms product from the lower section. This configuration will be denoted as a complex column. Davydyan et al. (1991) utilised a minimum-reflux model to obtain an understanding of the process mechanism. With the number of trays in both sections considered fixed, such a model of the column has two variable parameters, the reflux and reboil ratios. The distribution of products in the product vessels for these varied parameters was considered. An additional degree of freedom was also considered, in which the vapour flow rates in the column sections differ, brought about by the addition (a reboiler in middle vessel) or removal

(condenser in middle vessel) of heat from the middle vessel (Davidyan & Kiva, 1994).

For conventional batch columns (rectifier and stripper column configurations) the qualitative characteristics of the phase portrait (the distribution of product throughout the column) do not depend on any specific parameters, like reflux or reboil ratios, and the temperature in the reboiler (batch rectifier) or condenser (batch stripper) changes monotonically during operation (Davidyan & Kiva, 1994).

The complex column combines a batch rectifier and a batch stripper into a single distillation column. Davidyan & Kiva (1994) found that the phase portrait (the distribution of product throughout the column) of the complex column depends significantly on parameters such as the reboil and reflux ratio. The temperature in the middle vessel may not change monotonically during the process. Withdrawing no bottom product from the reboiler, but allowing distillate production, causes the complex column to behave as a batch rectifier - the temperature in the middle vessel rises monotonically. Setting the distillate product flow rate to zero and withdrawing bottom product, causes the column to behave as a batch stripper - the temperature in the middle vessel decreases. Davidyan & Kiva (1994) indicated that in some sense the complex configuration may be considered as a continuous distillation column with variable feed composition, with the middle vessel being a tray with a large hold-up (Davidyan & Kiva, 1994).

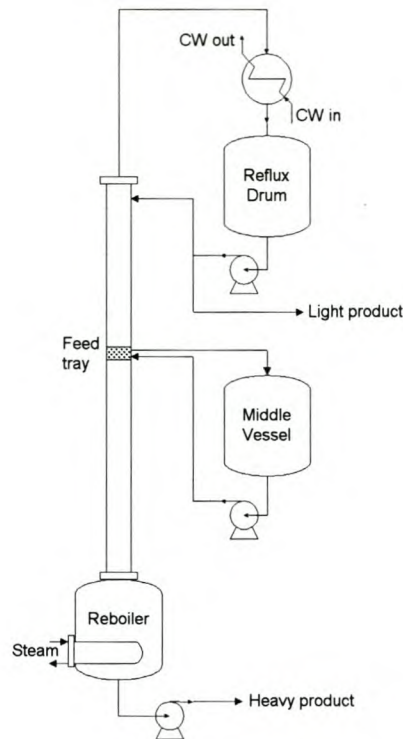


Figure 6-1 - Complex batch distillation column configuration.

Barolo et al. (1998) concluded that the dynamic behaviour of the complex batch column is far more complicated than that of conventional batch strippers and rectifiers. Interactions between operating parameters were found to exhibit unexpected plant responses during experimental *evaluation*. Several practical operation difficulties are posed by the configuration in figure 6-1. The liquid composition on the feed tray and in the middle vessel needs to be maintained at similar compositions, as to avoid poor separation performance. The recycle stream between the feed tray and the middle vessel, thus needs to have a sufficiently high flow rate to ensure similar compositions between the two sections. This presents operational difficulties as the high withdrawal rate required to maintain similar compositions in the two sections, may cause flashing in the suction line to the recycle pump which results in cavitation. Subcooling of the liquid in the suction line may be required to maintain sufficient net positive suction head for proper pump operation. The middle vessel content also needs to remain uniformly mixed, requiring agitation of the liquid content (Barolo et al., 1996).

The configuration of the complex column is, however, advantageous as it offers the opportunity to increase plant flexibility at low capital cost. A pre-existing continuous distillation column may be used for both conventional continuous and complex batch operations, provided that sufficient space is available for the installation of piping and valves (Barolo et al., 1996).

6.1.2 Multi-Effect Batch Distillation Column

For the simultaneous separation of more than three components in the complex column, an additional intermediate vessel may be installed at an appropriate column height. As the number of components increase such a multi-feed vessel complex column exhibits greater operational complexity (Hasebe et al., 1995).

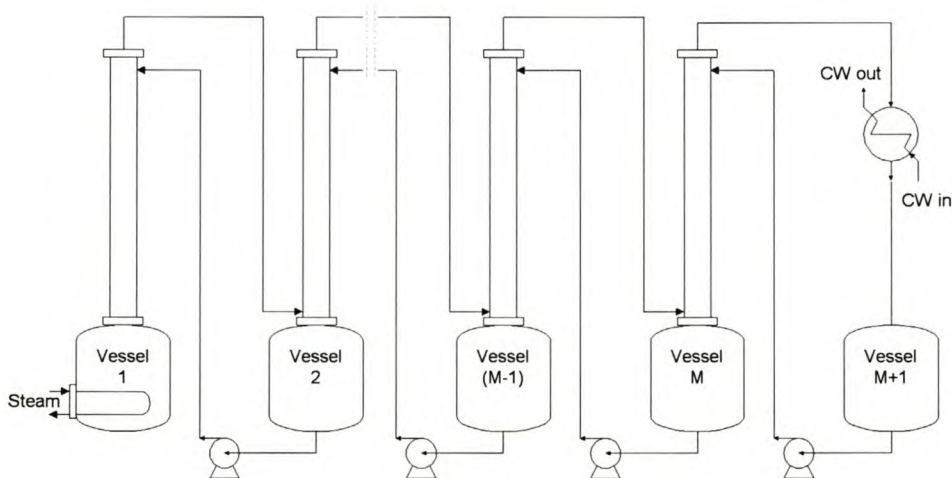


Figure 6-2 - Multi-Effect Batch Distillation system

Hasebe et al. (1995) showed through numerous simulations involving various conditions, that the operation procedure which has no withdrawal steps of products, shows better separation performance than for an operation procedure that included product withdrawal steps. Each component should be accumulated in a vessel during the whole separation period.

The system illustrated in figure 6-2 consists of a set of batch distillation columns connected sequentially and separates $(M+1)$ components with M columns simultaneously. Only the first column has a reboiler and only the final column has a

condenser. The vapour leaving from the top of a column is fed directly to the space between the tray section and the bottom vessel of the next column. The liquid in the bottom vessel of each column is returned as reflux to the top of the previous column. Through continued total reflux operation, the i 'th heaviest component is accumulated in the vessel of column i . The operation is terminated once the compositions of the liquid products in all the vessels reaches the desired specification.

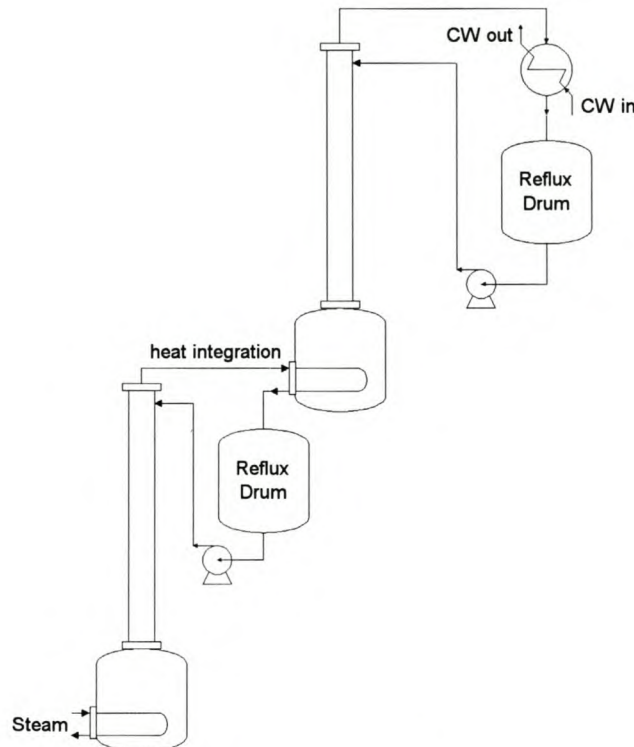


Figure 6-3 - Heat integration of batch distillation columns

Figure 6-3 illustrates a conventional heat-integrated batch distillation column. In this system, the condenser of the first column is heat integrated with the reboiler of the second column. The vapour from the top of the first column may be used as the heat source of the second column, provided that the first column is pressurised. For the system illustrated in figure 6-2, the vapour from the top of a column is fed directly to the next column. The latent heat of the vapour may be used effectively without giving consideration to the temperature difference between the heat source and the heat sink. The heat supplied to the reboiler is effectively used M times to generate the vapour of M columns. This heat integration results in greater energy efficiency for the separation of multi-component mixtures (Hasebe et al., 1995).

The *MEBAD* configuration overcomes some difficulties associated with the operation of the complex batch column (section 6.1.1). The total reflux operation *policy* is simpler than the product withdrawal *policy* of the complex column. The large recycle flow rate required to maintain similar compositions between the feed tank and the feed tray, is not required (Hasebe et al., 1995).

Barolo et al. (1998) indicated that the total reflux operation is only applicable for columns which have sufficient stages with respect to the separation objective. Also, the vessel capacities should be large enough to allow for the accumulation of a product or impurity.

6.2 PROCESS APPLICATIONS

The following subsections detail areas of application considered in the academic literature for the novel batch configurations presented (section 6.1).

6.2.1 Separation of ideal multi-component mixtures

Hasebe et al. (1992) considered the utilisation of the multi-effect column for the removal of both light and heavy impurities from a feed mixture. In the separation of multicomponent raw materials, there are many cases where the product is obtained by removing light and heavy impurities. Should a batch rectifier column be used to remove heavy impurities from the feed charge, a large amount of product must be removed as distillate to purify the essentially light product feed. More appropriately, such a removal of heavy impurities may be performed in a batch stripping column. Transversely, should a stripping column be utilised for the removal of light impurities, a large volume of product must be removed as bottoms in order to remove the light impurities. A rectification column is more appropriate for this application.

Hasebe et al. (1992) indicated that the simultaneous removal of light and heavy impurities may be executed conventionally in a combination of a batch rectifier and a batch stripper.

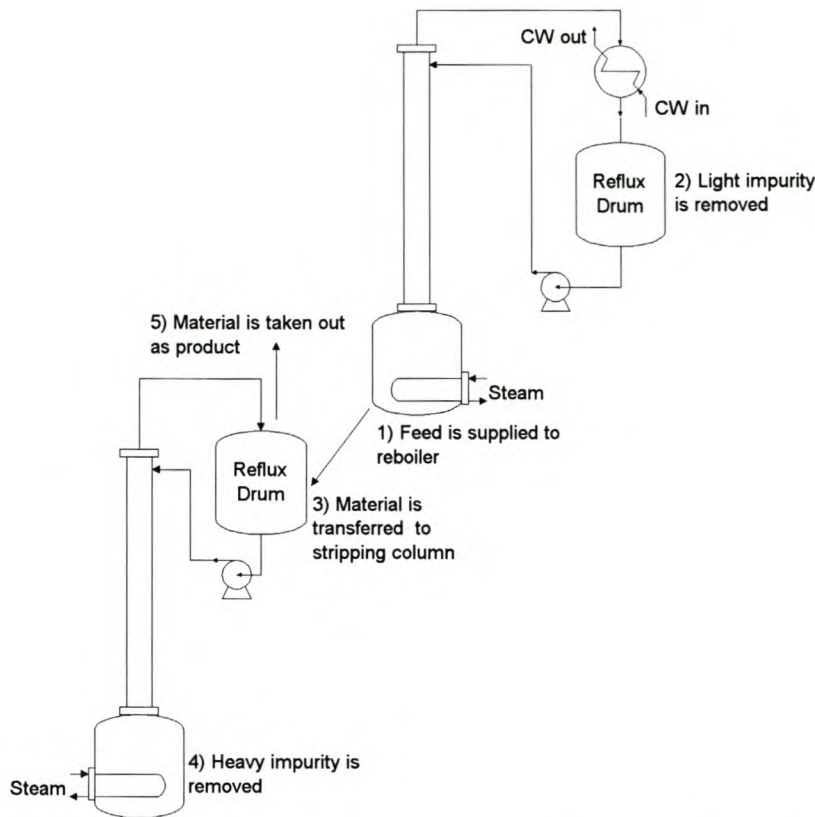


Figure 6-4 - Conventional sequencing for the separation of light and heavy impurities.

By using ordinary and stripping batch distillation columns successively in a conventional approach (figure 6-4), light and heavy impurities may be removed without vaporising a large amount of product. The feed is supplied to the reboiler of the ordinary column and the light impurity is withdrawn from the top of the column. Once the light impurity has been removed the material is transferred to the reflux drum of the stripping column and the heavy impurity is withdrawn from the bottom of the stripping column. The material in the stripping column reflux drum is removed as product (Hasebe et al., 1996).

Should the separation be executed cyclically using a rectification and a stripping column, the condenser of the stripping column may be heat-integrated with the reboiler of the batch rectifier (figure 6-3). Pressurisation of the stripping column and chemical transfer may be avoided by utilising the complex or multi-effect batch distillation configuration (section 6.1). The *MEBAD* or complex configuration thus offers a considerably simpler operation strategy and less costly operation (Hasebe et al., 1996).

6.2.2 Extractive Batch Distillation

To facilitate the separation of azeotropes, extractive distillation is typically utilised. In extractive distillation a heavy component, entrainer, is fed close to the top of the column. This component changes the relative volatilities of the azeotropic species and entrains some of the components down the column that normally accumulate in the distillate.

Safrit & Westerberg (1995) showed that it is possible to recover 100% of the distillate product of an azeotropic mixture (methanol, acetone & water) using a batch rectifier and a complex column. However, the rectifier (conventional batch column) requires a still of infinite size, while the complex column does not have this limitation. The rectifier requires an infinite size still pot as water is continually added to the system but never removed. The complex column configuration allows for the middle vessel to be "steered" towards the composition of the intermediate product, affecting a three-component azeotropic separation in one column with no waste. The product withdrawal rates or vessel hold-ups determine the direction in which the middle vessel composition is steered. Using still path steering for the complex column, an infinite still is not required as water is taken from the reboiler and recycled as entrainer (Safrit & Westerberg, 1997).

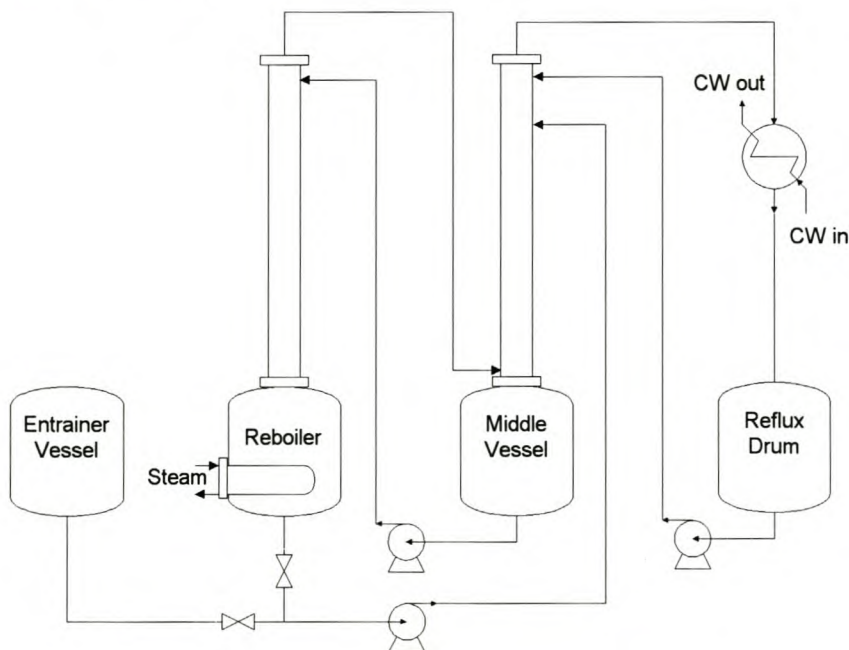


Figure 6-5 - Extractive batch distillation utilising a *MEBAD* column

6.2.3 Reactive Batch Distillation

Mujtaba & Macchietto (1994) considered a theoretical comparative study with conventional and unconventional columns for simultaneous chemical reaction and distillation, with the following reversible reaction scheme. The relative volatilities of the components, with respect to the light key, are $\alpha_{iD} = \{0.56, 0.50, 0.44, 1.0\}$.



The reaction products are C (main product) and D, with D being the most volatile component and C being the least volatile component in the reaction mixture. Separation of the product by distillation permits increased conversion, while also yielding the product in concentrated form.

The equilibrium conversion for the considered chemical system, in the absence of distillation, is 58.5%. To obtain the desired purity in C with a mole fraction of 0.7 [] in the bottoms, one or both of the products need to be removed from the reboiler as vapour.

Table 6-1 - Reactive batch distillation comparative study

<i>Column</i>	<i>Conversion</i> [%]	<i>Product</i> [kmol]	<i>Reflux</i> Ratio	<i>Reboil</i> Ratio
Complex	70.9	1.52	0.899	0.952
Rectifier	78.4	2.78	0.934	-
Stripper	61.3	1.29	-	0.956

Reflux ratio is the selected control variable for the conventional column and reboil ratio for the inverted column. Both are used as control variables for the complex column.

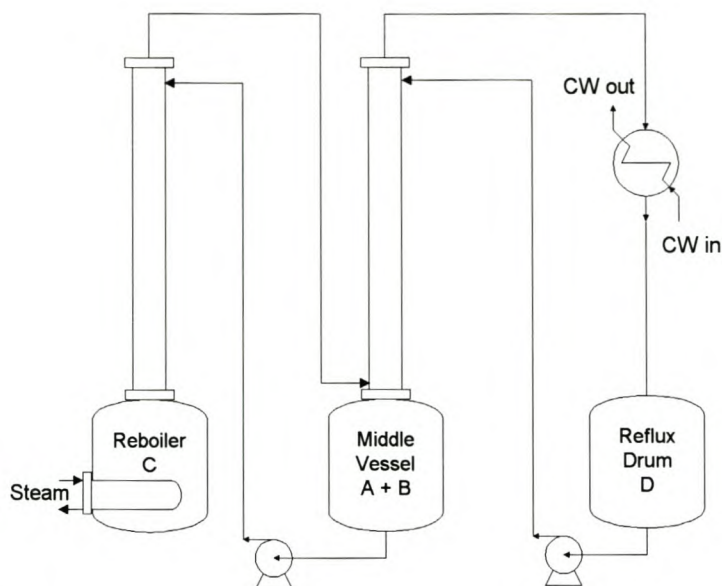


Figure 6-6 - Batch Reactive Distillation in a MEBAD column as proposed by Mujtaba & Macchietto (1994)

The comparative analysis results are presented in table 6-1. The conventional column is more efficient for the considered chemical system, as D is a great deal more volatile than either A, B or C allowing for ease of separation. The loss of reactant A for the rectifier column in the distillate is minimal (3%). For the stripper column conversion is lowest as the separation of C from A & B is more difficult, as their relative volatilities are similar. The loss of reactant A (4%) for the stripper column is also higher than in the rectifier column. The loss of reactant A was found to be far greater at 18% in the complex column, but as both end products are removed simultaneously good conversion and amount of distillate product is achieved. Performance of the complex column is between the rectifier and the stripper configurations for this feed composition (Mujtaba & Macchietto, 1994).

6.3 ECONOMIC POTENTIAL OF THE MULTI-VESSEL COLUMN

The *MEBAD* column's primary energy feature is that the energy to separate the mixture is considerably less than in conventional batch distillation due to the heat integration of several column sections (section 6.2.1). Hasebe et al. (1995) indicated that the energy efficiency of a multivessel batch column could be comparable to conventional continuous distillation column sequencing. Hasebe et al. (1995) found that the separation performance of the *MEBAD* system approach that for a continuous direct sequence configuration as the number of components for separation increased.

6.3.1 Conventional batch distillation operating strategies

The optimisation of a rectifier column over the entire batch period leads to a optimal time control problem formulation. Results from optimisation studies have shown increased performance of conventional columns using an optimal reflux policy, compared to constant reflux and constant composition policies. Time and energy savings in the order of 20% are obtainable, depending on the mixtures to be separated. The implementation of an optimal reflux policy is as yet not widely applied in industry. Industrial batch columns are operated most frequently with a simple reflux policy, that is, the number of off-cuts are predefined and the reflux ratio is constant during operation. For multicomponent mixtures a constant reflux policy requires that a large number of slop cuts need to be taken, analysed and blended to obtain the appropriate products. It is not optimal to maintain a constant reflux policy which results in longer production times with higher energy cost (Wittgens & Skogestad, 1998).

The constant composition reflux policy entails a feedback control scheme and is generally more energy efficient. Composition is frequently inferred (due to the expense and time delay of on-line composition measurement) by maintaining a constant temperature at a given location in the column, thus indirectly controlling the top product composition. Change-over between products occurs once the reflux flow increases above a predefined limit. An off-cut product or fraction may be produced, with the switch to the second product starting once the next temperature set point is

reached. Off-cut collection is ceased once a change-over temperature determined from the boiling point of the next product fraction, is reached. This change-over temperature is slightly above the temperature setpoint of the next product. Most of the lighter component may so doing be removed from the column before the next product is collected. Operation during off-cut collection is most frequently completed in the open-loop. This procedure is repeated until $N_c - 1$ product fractions are withdrawn over the top, with the heaviest product being withdrawn from the reboiler. Blending of slop-cuts into the products may follow once all the products have been collected, to consequently maximise the production rate (Wittgens & Skogestad, 1998).

6.3.2 Multi-effect batch Distillation

The simplest strategy for operating the multi-effect batch distillation column is a total reflux policy (section 6.2.1). This policy is advantageous compared to conventional batch column operation policies, in that no product change-overs are required during operation and due to the heat integration of the sequenced columns.

Skogestad et al. (1995) compared the energy usage between rectifier batch distillation and the multi-effect batch distillation column. The same volume of initial charge and boil-up ratio (Total reboiler hold-up / vapour flow rate) was considered. The number of stages in the rectifier batch column was equated to the sum of all the stages in the multivessel column. A constant reflux policy was adopted for the rectifier batch column. The results indicate that a rectifier column operated at 85% of total reflux, required 70% more energy and produced approximately 10% off-cut in order to meet all four product specifications. With an internal reflux ratio that produced a minimum amount of off-cut, approximately 150% (1.5x) more energy is utilised.

Wittgens & Skogestad (1998) also compared the multivessel column to a constant composition reflux policy in a conventional batch column. The same volume of initial charge of a four component system was considered, with the same total number of stages in the two batch column configurations. For the rectifier column, no slop-cuts were blended into the products. Blending the off-cut products into the over specified products, resulted in a net increase in the production rate of less than 10%.

For multi-vessel batch distillation columns the off-cut is negligible, with the remaining column hold-up being the only significant off-cut produced. The separation of the feed mixture was performed in a significantly shorter batch time and thus energy consumption for the *MEBAD* column. For columns of identical length the specific considered separation consumed approximately 30% less energy in the *MEBAD* column, compared to the rectifier column. As no off-cut is produced in the *MEBAD* column an increase in production rate in the order of 35% is evident.

Although the two reflux policies considered for the rectification columns were far from the optimal reflux policy, these policies reflect common industrial practice. Mujtaba & Macchietto (1994) indicated that a comparative analysis between the performance of the rectifier and complex columns, depends significantly on feed composition.

6.4 OPERATION AND CONTROL POLICIES FOR THE MEBAD CONFIGURATION

Batch distillation offers increased flexibility over conventional batch distillation columns and a variety of operating modes. Two conventional operating modes include (1) constant reflux and variable product composition and (2) variable reflux and constant product composition of a key component. The variable reflux mode is the only candidate for closed-loop composition control. Optimal reflux policy represents the third mode of operation which is neither constant reflux nor constant product composition and generally involves an open-loop control problem. With the advent of novel batch distillation designs, the number of operating modes has increased substantially. The middle vessel column may be operated with even great flexibility than the conventional columns, as the degrees of freedom are far greater. (Farschman & Diwekar, 1998)

6.4.1 MEBAD total reflux operation with vessel hold-up control

Hasebe et al. (1995) proposed a level control strategy for the multi-effect batch distillation column in which the hold-up of each vessel is calculated in advance by taking into consideration the feed quantity, feed composition and product specification. After feeding the calculated amount of raw material to each vessel, a total reflux operation policy is followed until the composition in each vessel matches the prescribed specification. By using a level control scheme, the hold-up in each vessel is held constant.

Should a discrepancy exist between the measured feed composition and the actual composition, products may not satisfy their specifications, despite keeping the hold-up of each vessel constant. This results as no negative feedback is provided for measurement errors (Hasebe et al., 1995).

Hasebe et al. (1995) proposed a cascade control system for each vessel to control each vessels' composition (figure 6.7) in the presence of composition uncertainty. The hold-up of each vessel is selected as the *manipulated variable*.

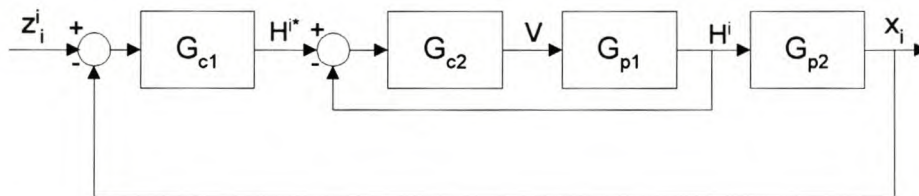


Figure 6-7 - Cascade control for the MEBAD configuration proposed by Hasebe et al. (1995).

Should the composition of the dominant component in product i (x_i) does not meet the desired specification (z_i^j) due to inaccurate final hold-up estimation at a given time (figure 6-7), the set point of the hold-up of the vessel i (H^{i*}) is reduced.

For the suggested mode of operation, the composition in all vessels are the same as the feed composition at the beginning of the operation. Executing composition control from start-up, would thus cause the vessel hold-ups to have severe fluctuations.

Hasebe et al. (1995) proposed that cascade control be initialised once the product composition satisfies one of the conditions listed in table 6.2. During the initial stage of operation, the composition in the vessel changes slowly. The third constraint (table 6.2) is introduced to avoid the activation of cascade control at the beginning of the operation.

Table 6-2 - Conditions for initialising the cascade control scheme

<i>Conditions for initialising cascade control scheme</i>	
<ul style="list-style-type: none">• $x_i > z_i^i$• $\frac{dx_i}{dt} < 0.01 \left[h^{-1} \right]$• $\frac{d^2x_i}{dt^2} < 0$	

The operation policy of maintaining a constant hold-up in each vessel is, however, not optimal in terms of minimising the total batch time. Minimising the total batch time presents an optimal control problem, which may regard the hold-up of each vessel as time-variant. Hasebe et al. (1997) compared two optimisation policies. The constant hold-up policy entails the optimisation of the initial vessel hold-ups as to maximise the desired performance index (production rate). During the operation the hold-up in each vessel is held constant at the predefined calculated optimum value. The variable hold-up policy entails optimising each vessel's hold-up as a function of time. The initial charge to each vessel is also optimised. The benefit of such calculated optimal hold-up profiles is, however, dependent on the accuracy of the dynamic model of the plant. Hasebe et al. (1997) demonstrated that the variable hold-up policy is superior to the constant hold-up policy. The introduction of negative feedback to compensate for modelling errors remains paramount in the face of uncertainty.

6.4.2 Temperature control feedback structure

All current batch distillation operating policies may be realised in a *MEBAD* configuration. In the most general configuration (section 6.1.2) the vessel hold-ups and product flows represent $2N_c - 1$ degrees of freedom. The simplest operation policy is that of total reflux operation, in which the N_c product rates are set to zero (section 6.1.2). A total reflux policy is superior to the conventional batch distillation column operation policies, as total reflux always allows for the maximum possible separation performance. Firstly, no product change-overs are required which substantially simplifies the operation. Secondly, the energy requirement may be less due to the multi-effect nature of the operation (Wittgens et al., 1996).

For the separation of multicomponent feeds, Wittgens et al. (1996) proposes a feedback control structure based on $N_c - 1$ temperature controllers. The reflux flow in each of the upper $N_c - 1$ vessels is adjusted by temperature controllers at an appropriate location in the preceding column section. The hold-up in each vessel is thus adjusted indirectly by the varying reflux flows to meet the temperature specifications. No level controller or level measurement is incorporated, although the use of upper and lower level switches may be foreseen for safety reasons. The set point for each temperature controller may be set appropriately to obtain two pure components at the top and bottom of the particular column section. The set points may be calculated from steady state conditions to achieve the desired product. The set points may also be optimised as a function of time.

6.4.2.1 Experimental verification

Wittgens et al. (1996) verified the simulation results experimentally in a laboratory scale unit for the separation of methanol, ethanol, n-propanol and n-butanol. The results indicate that the same steady state compositions are achieved in the vessels, regardless of the initial charge composition (only vessel hold-ups differ at steady state). This results as the number of temperature controllers equals the number of degrees of freedom at steady state. The temperature profile in the column at steady state is thus determined, assuming that multiple steady states do not exist.

For the experimental verification, each reflux controller's set point was set to the arithmetic mean of the boiling points of the components expected to accumulate in the two adjacent vessels. Standard PI-controllers were utilised with slow controller parameters, to avoid extensive control action during start-up and in the presence of disturbances. The start-up policy involved charging the entire feed to the reboiler. Once condensate started to collect in the uppermost vessel, reflux was introduced to the column sections. Once the temperature in the uppermost column approached its set point, the temperature controller for reflux to that column section is activated. The vessel content from which the reflux is pumped, varies during operation and is thus indirectly controlled by the temperature controller.

Similarly, after establishing (in turn) a minimum hold-up in the intermediate vessels reflux is introduced to the preceding column section. Reflux back to the reboiler is eventually established. During start-up the volumetric flow rates prescribed by the controllers are limited to avoid emptying the reboiler and the accumulation of light component in the lower vessels. After establishing stable flows in the column, the controllers for the intermediate columns are activated. Distillation is continued until temperature and reflux changes are small for a given period of time.

6.5 EXPERIMENTAL

6.5.1 Experimental Set-up

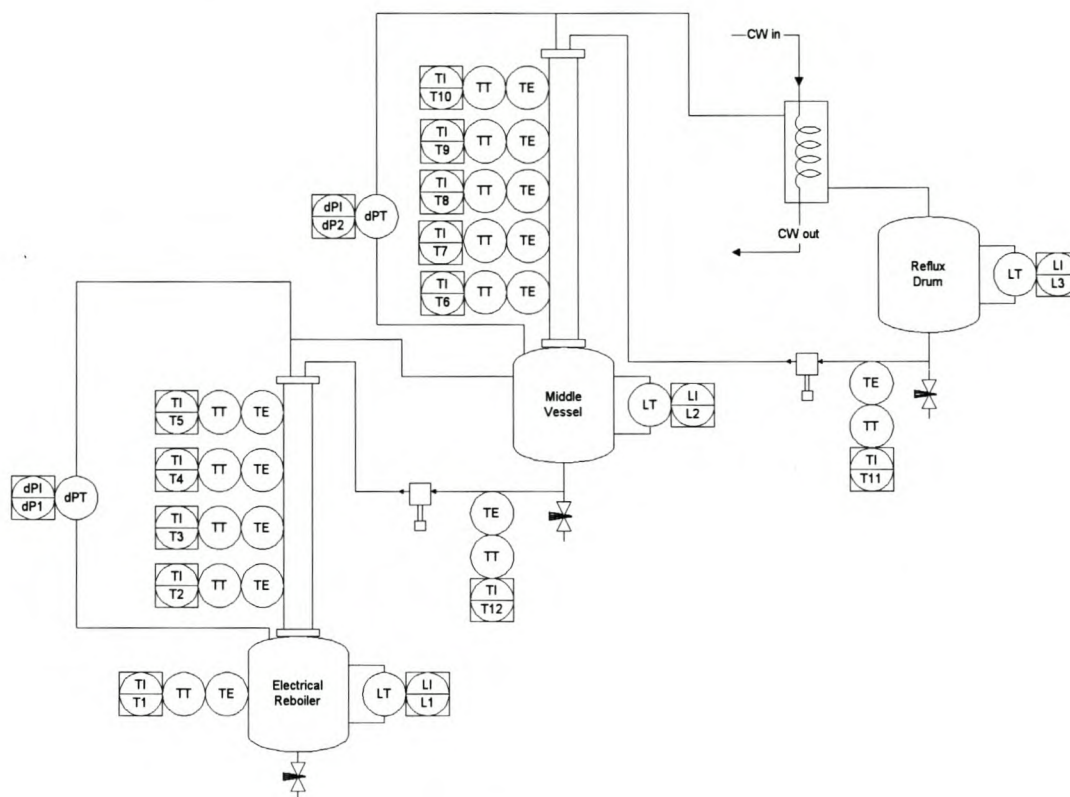


Figure 6-8 - Illustration of experimental setup

In order to demonstrate *SANE*'s ability to learn control behaviour from a real system, a Multi Effect Batch Distillation (*MEBAD*) column was constructed on a pilot plant scale. The following sub-sections detail the process layout as illustrated in figure 6-8.

6.5.1.1 Mechanical Equipment

The *MEBAD* design for a ternary system requires the utilisation of two column sections with a middle vessel. Two stainless steel (schedule 10) columns with an inner diameter of 0.068m were utilised, with Montz BSH 400 as structured packing. Each packing segment has a length of 0.205 [m], with each packing segment having a 90° orientation to the following segment. This allows for effective liquid redistribution in

the column. The length of the first column is 2.0m with 1.90m of structured packing. The second column has a length of 2.6m, with 2.5 m of structured packing.

High performance orifice distributors were designed and installed in the top of each column. Each distributor has four 3mm ID drip points, with each perforation equipped with a short, flow straightening tube. This eliminates lateral movement of the liquid jets due to jet instability and prevents the liquid from running along the bottom of the distributor pan. Each distributor has five vapour risers to allow for vapour passage without liquid entrainment and to ensure minimum pressure drop over the distributor. No internal redistribution was provided as the structured packing was deemed to have sufficient redistribution characteristic (wall wipers on packing) to prevent maldistribution.

A 50L spherical glass vessel serves as reboiler for the batch distillation column. The glass reboiler allows for a maximum operating pressure of 1.5 bar(a). The glass middle vessel has a capacity of 30L, with the glass reflux drum having a capacity of 20L. Each vessel is provided with a 4mm valve to allow for gas chromatographic samples to be taken.

A glass coiled vertical condenser provides a maximum cooling capacity of 3 kW. Cooling water is supplied from a cooling tower. The condenser pressure is maintained at atmospheric pressure through a water seal.

Two positive displacement pumps with a maximum volumetric flow rate of $60 \frac{1}{h}$ and a maximum discharge pressure of 14 bar(a) serve as reflux pumps for column. The adjustable piston stroke length was set to allow for a maximum volumetric flow rate of $35 \frac{1}{h}$. Variable speed drives are utilised to allow for variable reflux rates to the column sections. A frequency of 50 Hz thus corresponds to a flow rate of $35 \frac{1}{h}$ on a linear scale. The gearing between the motor and pump was also selected to allow for a stroke frequency of 56 strokes per minute at 50 Hz to reduce the probability of cavitation. For both reflux pumps sufficient suction head was provided to eliminate

the probability of cavitation. Also $\frac{1}{2}$ " stainless steel tubing was used for the suction piping, to reduce the liquid velocity in the suction line. $\frac{1}{4}$ " stainless steel tubing was used for the discharge piping, as to reduce the liquid residence time in the discharge piping, thereby reducing the degree of reflux liquid subcooling.

6.5.1.2 Instrumentation and signal processing

The experimental arrangement is provided with a number of sensors to facilitate accurate process state *evaluation* and troubleshooting. To ensure minimal *environmental* interference and thus reduce sensor noise, instrument cabling with mylar shielding is utilised throughout. Possible electromagnetic interference is also eliminated by running the instrument and electrical cabling along separate cable trays.

Twelve PT100 temperature sensors are installed at various locations in the experimental setup (figure 6-8). Inhead transmitters for signal conversion are installed in the RTDs to ensure minimal signal interference along instrument cable lines. The inhead transmitters convert the PT100 signal output to a 4-20 mA signal. The temperature sensors provide a means to infer the chemical composition at the particular location.

A differential pressure sensor is installed for each vessel to gauge the vessel level from the hydrostatic pressure. Sensym SCX C solid state sensors with a range of 0 - 1 [psi] provide the necessary signal processing. Each column is also provided with a differential pressure sensor to provide an indication of column vapour velocities and to serve as a means of troubleshooting possible column flooding difficulties. The Sensym SCXL004DN solid state sensor with a range of 0 - 4" H₂O provides for the column differential pressure signal processing. The mV output from each Sensym differential pressure sensor is converted to a 4-20 mA signal utilising the Burr-Brown Corporation's XTR104 microchip.

Variable speed drives serve to control the reflux rate to each column section. A 0-10V output signal to each variable speed drive determines the 0-50 Hz motor frequency. An advantage of utilising a pump and variable speed drive configuration lies in that the transfer function from motor frequency to flow rate is linear, as opposed to a transfer function for a control valve which may be nonlinear.

A Pentium class personal computer (PC) is installed in the control panel. To avoid electromagnetic interference from particularly the variable speed drives, a steel plate separates the PC from the field instrumentation and electrical components. The PC is fitted with two Advantech 812PG analogue I/O PCI cards. Each analogue 4-20 mA analogue input signal is converted to a 1-5V signal over a 0.1% 250 Ω precision resistors, which is consequently converted, with a 12 bit precision, by the 812PG analogue-to-digital channels for digital processing. The 0-10V analogue output signal to the variable speed drives is also provided from the 812PG I/O cards. The MMI (section 6.5.1.4) interfaces with the plant instrument signals through the 812PG I/O cards. The sampling frequency of the Advantech I/O cards is set to 0.25 [s].

To reduce the impact of measurement noise, a median filter (nonlinear filter) is utilised to filter any A/D conversion errors or instrument noise. The past four and the current signal readings for each sensor are used in a median calculation, with the median value of the 5 readings taken as the actual reading. This method is deemed applicable as the process time constants are far greater than the time frame of 1.25 [s] used for signal filtering.

6.5.1.3 Electrical Equipment

The content of the 50L glass reboiler is heated utilising a heating mantle with a maximum heating capacity of 2.93 kW. The heating mantles output is, however, limited to 2.2 kW to protect the glass vessel against an excessive temperature gradient across the glass wall. An additional 1.5 kW stab-in heating element is also installed in the reboiler vessel. The heating duty to each heating element is controlled through the use of a solid state relay. Electrical power is supplied to the heating element over a fraction of a user defined cycle time. The heat duty to each element is controlled by

changing the fraction of time for which electrical power is supplied through the use of a solid state relay.

6.5.1.4 Man Machine Interface (MMI)

A Man Machine Interface for the experimental setup was developed utilising Borland's Delphi 4.0. A user friendly interface is provide for access to *process variables* via a Windows Explorer type hierarchical tree view (figure 6-9). Each *process variable's* real-time value and its historical trend may be accessed by clicking on the desired folder in the treeview. The real-time value is indicated in the statusbar, while the historical trend is displayed on the graph. Two *process variables* may be displayed concurrently for analysis and troubleshooting purposes. All *process variables* are logged to a Borland Paradox 7.0 database with a user defined frequency.

A tabsheet is also provided for PI controller response *evaluation* and tuning. For PI controller *evaluation* each controller's real-time set point, *process variable* and *manipulated variable* is displayed on the graph panel. This graphical output may consequently be utilised for tuning the PI controller parameters.

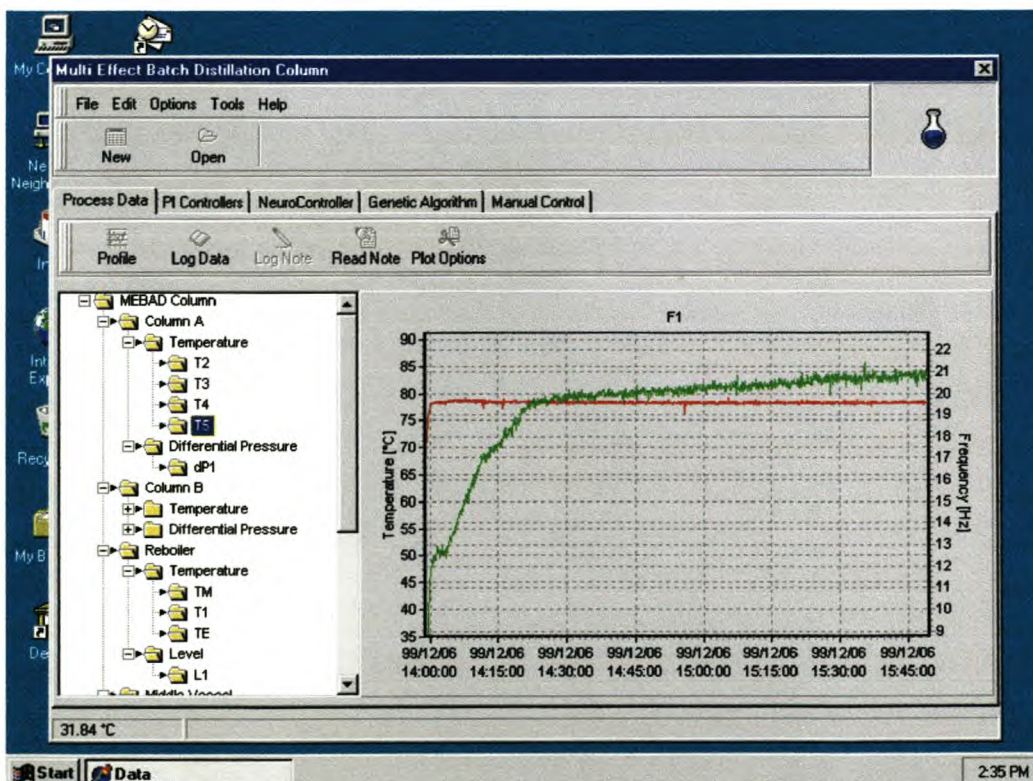


Figure 6-9 - Delphi Graphical User Interface utilised as Man Machine Interface.

6.5.2 Experimental Procedure

For experimental verification of the applicability of the *SANE* algorithm in real world applications; the separation of methanol, ethanol and n-propanol was considered in the *MEBAD* column described in section 6.5.1.

6.5.2.1 PI control experimental procedure

To facilitate the comparison between the temperature control approach proposed by Wittgens et al. (1996) (section 6.4.2), the experimental setup was arranged to allow for PI control of a temperature set point (table 6-4) in each column (figure 6-10). The temperature sensors at the top of each column were selected as *process variables*, as the time delay in the temperature response to a change in reflux rate is shortest at the top of each column. The final control element for each temperature controller is the variable speed drive, which adjusts the frequency to the pump motor, thus manipulating the reflux flow rate to each column. No constraints were applied to the PI controller output (section 6.4.2), as the large reboiler capacity provided sufficient damping against draining the reboiler content. Also, the temperature and vessel level responses during start-up with PI control were deemed satisfactory.

Table 6-3 - MEBAD column temperature set points

<i>Process Variable T5 (PV1)</i>		<i>Process Variable T10 (PV2)</i>	
Set point	78.3 [°C]	Set point	69.0 [°C]

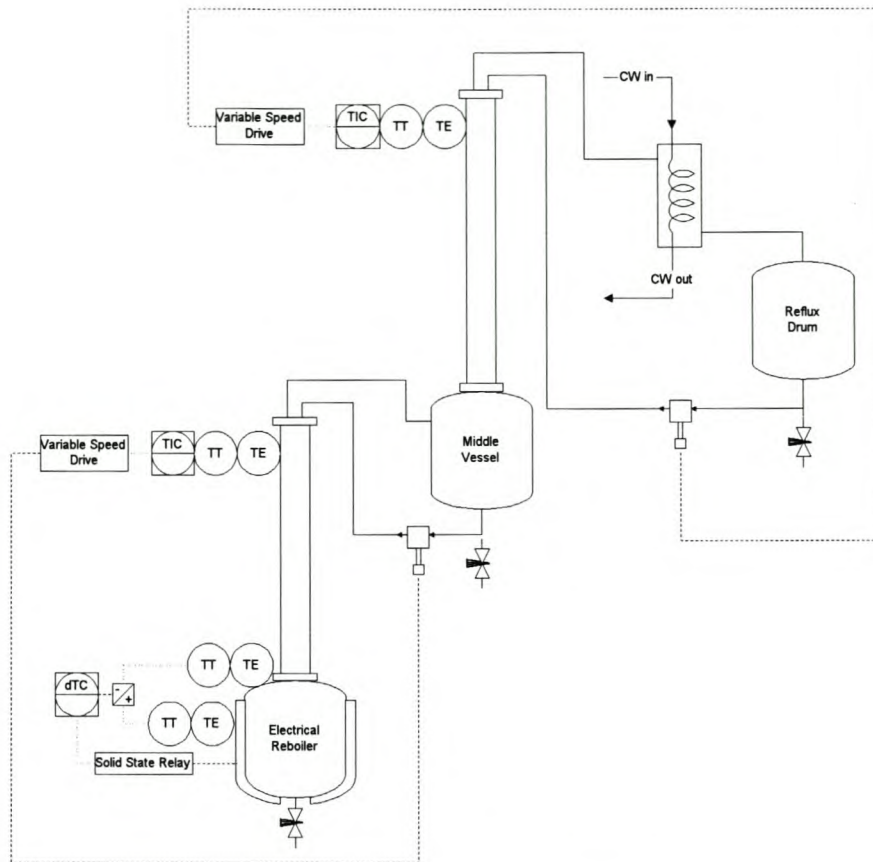


Figure 6-10 - PI Control implementation for the experimental setup

In order to ensure similar reboiler duties for different experiments, the heat duty to the reboiler vessel is also subjected to PI control. Assuming that the overall heat transfer coefficient remains approximately constant while the reboiler content is boiling, the temperature difference across the reboiler's glass wall may be used as *process variable*. The temperature between the heating mantle and the glass reboiler is consequently measured, and the reboiler liquid content temperature is subtracted. This differential temperature thus serves as the *process variable* for the PI controller. The final control element in this heat duty control arrangement, is a solid state relay which allows for fast non-mechanical switching of the power supply to the heating mantle. A unit heating period (cycle time) of 10 [s] was used, with the PI controller's *manipulated variable* being the fraction of the total heating period that power is supplied to the heating mantle. For example, a *manipulated variable* output of 7 [s], allows for 7 [s] of power supplied to the heating mantle and 3 [s] without power supply. This allowed for effective regulation of the heat duty to the reboiler by maintaining a constant differential temperature across the glass wall.

6.5.2.2 Neurocontroller learning experimental procedure

The *fitness* function criteria utilised for the *neurocontroller* learning trials on the *MEBAD* column, is identical to that used for the simulation studies. That is, a *ITAE* response is required for the given control objective.

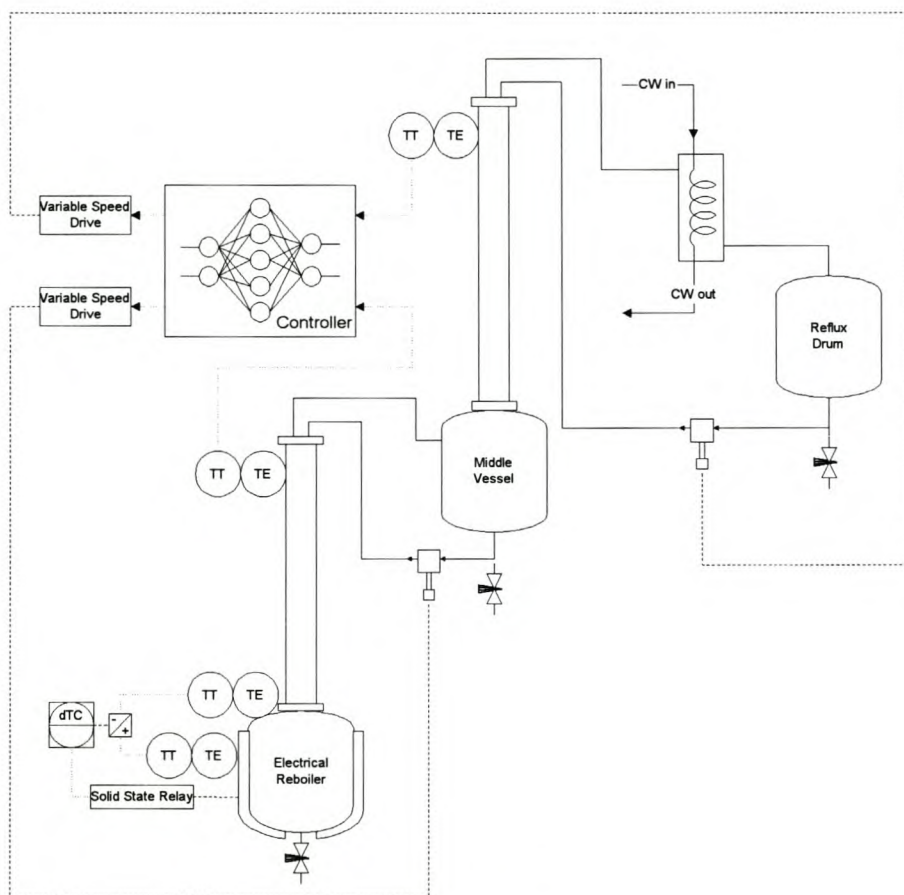


Figure 6-11 - Neurocontrol implementation for the experimental setup.

From the simulation studies presented in chapter 5, it is empirically evident that a single *evaluation* trial produces more beneficial learning information, should the initial plant condition for the trial be in a general vicinity of the desired plant state (set point). Also, it was empirically established that learning ability (speed) is enhanced, should the learning *agent* encounter both positive and negative deviations from the desired set point value. The *MEBAD* column was thus initially brought to the desired steady state (final column temperature and vessel composition profiles) utilising the PI control procedure outlined in section 6.5.2.1, before starting a learning sequence encompassing a number of generations.

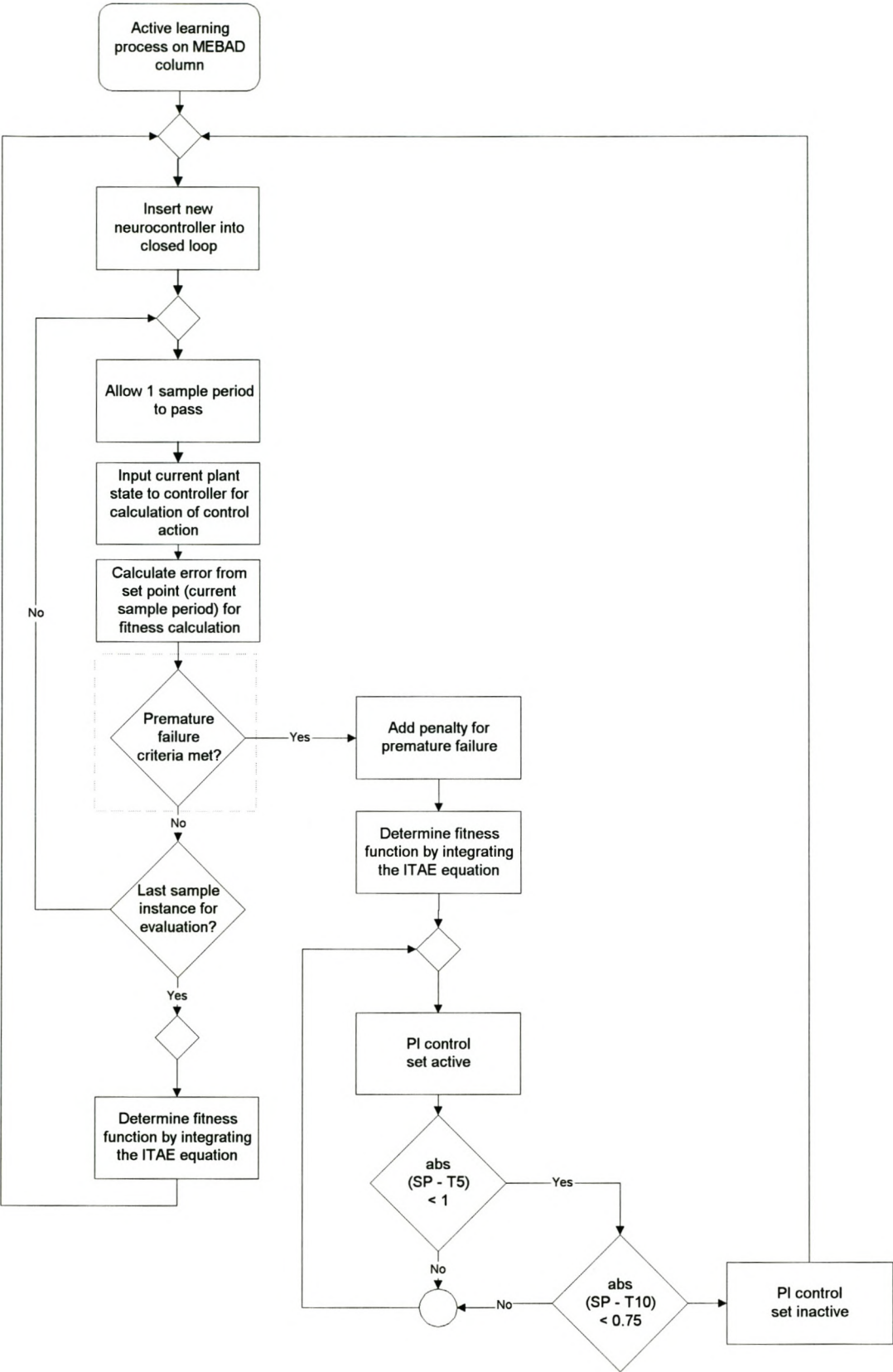


Figure 6-12 - Evaluation flow diagram for the MEBAD online learning process.

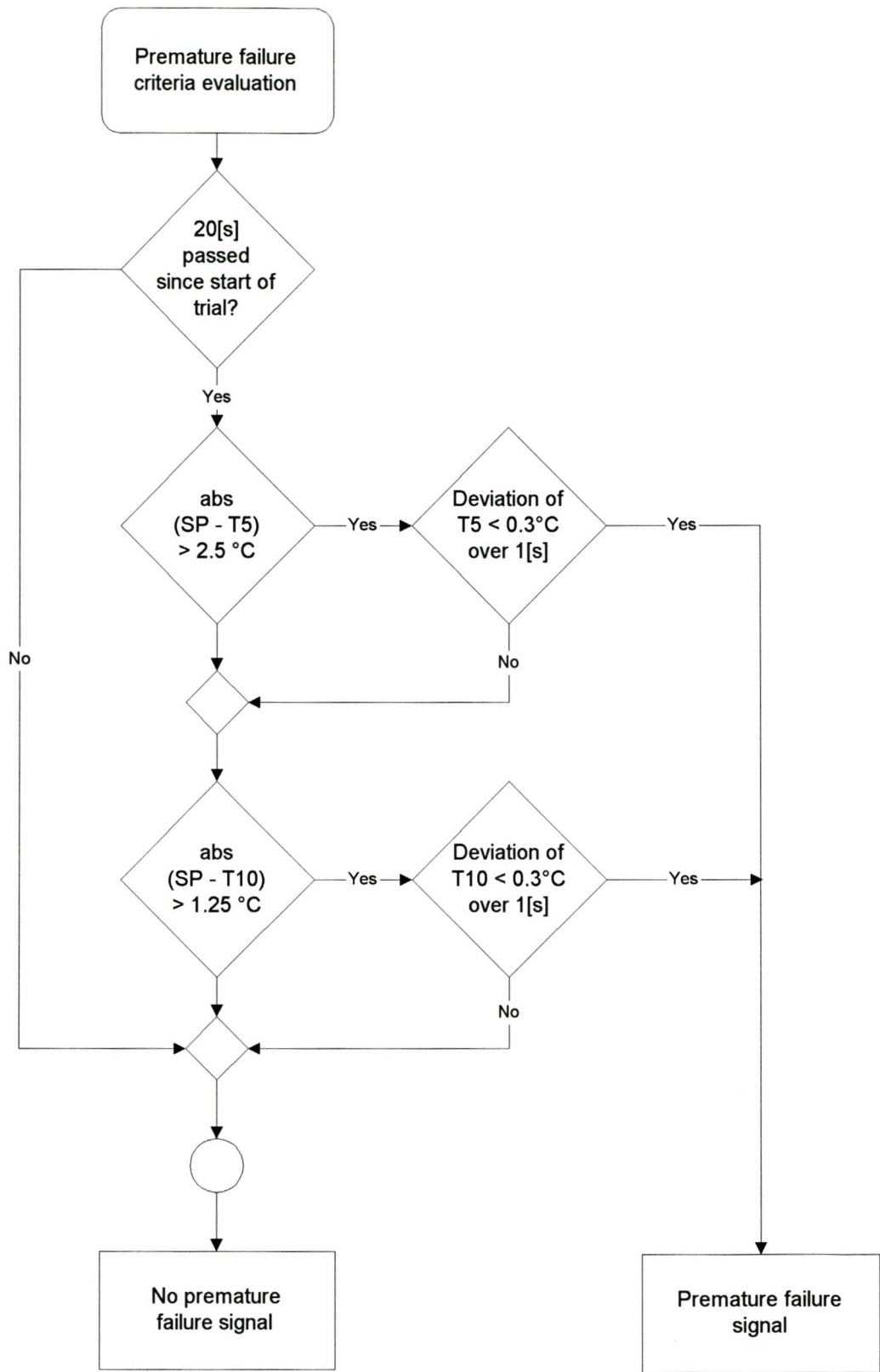


Figure 6-13 - Premature failure criteria breakdown encapsulated in dashed lines in figure 6-12.

The inputs to the *neurocontroller* were the two temperature set points and the real time temperature readings for the *process variables* (figure 6-11). The input signals to the input nodes were linearly normalised over a small temperature range. For example, for the *process variable* $T5$, 70 [°C] represented an input value of 0, with 85 [°C] having an input value of 1. This allows for maximum controller sensitivity to changes in the *process variables*. The output nodes determined the motor frequency of the two reflux pump motors. The possible output frequencies were limited to between 7 Hz (prevent the occurrence of zero reflux) and 35 Hz (avoid possible pump cavitation). The hidden layer was represented by 8 hidden *neurons*. The development of a single multi-input multi-output (MIMO) *neurocontroller* was thus the objective of the *evolutionary* algorithm.

In this study, the first generation (*neuron* and *network populations*) was initialized with random *weights* and connections. The process of *evaluation* for each neurcontroller is illustrated in figure 6-12 and figure 6-13. Each *neurocontroller* (learning *agent*) in the *population*, was afforded an *evaluation* period of 180 [s], in which to demonstrate its ability to control the *MEBAD* column. To prevent the *MEBAD* column from deviating too far from the final steady state due to poor *neurocontrollers* in the *population*, PI control was introduced during learning in a supervisory capacity. During an *evaluation* trial for a given *neurocontroller*, should the absolute value of the error from set point be greater 2.5 [°C] for $T5$ or greater than 1.25 [°C] for $T10$, a trial failure signal is generated and PI control is activated. Should the PI control reduce the absolute error for $T5$ to less than 1 [°C] and the absolute error for $T10$ to less than 0.75 [°C], PI control is deactivated and the next *neurocontroller* in the *population* is inserted into the closed loop. A hysteresis bandwidth is thus incorporated to ensure that an *evaluation* trial is not started on the failure boundary. In order to reduce the disruptive effects of sensor noise after changing from PI control to neurocontrol, a 20 [s] period is allowed for in which the *neurocontroller* can not fail regardless of the sensor readings. This allows for the re-establishment of a stable liquid film over the temperature sensor, which is significantly disrupted should the PI control action differ significantly from that proposed by the *neurocontroller*. Also, despite the utilisation of a median filter to eliminate signal processing errors, periodic incorrect temperature readings were obtained due to incorrect analogue to digital conversions. This could lead to a

neurocontroller trial failure, based on incorrect information. To counteract this eventuality, a temperature reading was required to deviate by less than 0.3 [°C] from the temperature reading taken 1 [s] in the past, before a failure signal was generated. This procedure was performed for all *neurocontroller evaluation* trials. The temperature history for *T5* and *T10* and the respective output frequencies to each pump motor, is illustrated in figure 6-14 and figure 6-15.

Once the entire *population* of *neurocontrollers* had been evaluated, the genetic *operators* were applied (section 3.6) and the next generation is consequently evaluated. The information which encodes the *neuron* and *network population* is stored in a Paradox 7.0 database table for each generation.

Should a *neurocontroller* fail to control the plant within the defined temperature bandwidth for each *process variable*, premature termination (failure) of a trial results (described above). Such *neurocontrollers* are penalised for premature failure, in that a maximum error is incurred for each remaining time step in the trial. The maximum error for the temperature *process variables* was selected as the full operating bandwidth range, with an additional penalty of 10%. This procedure ensures that *individuals* that are not able to keep the column within the given bandwidth are poorly ranked in the *population*.

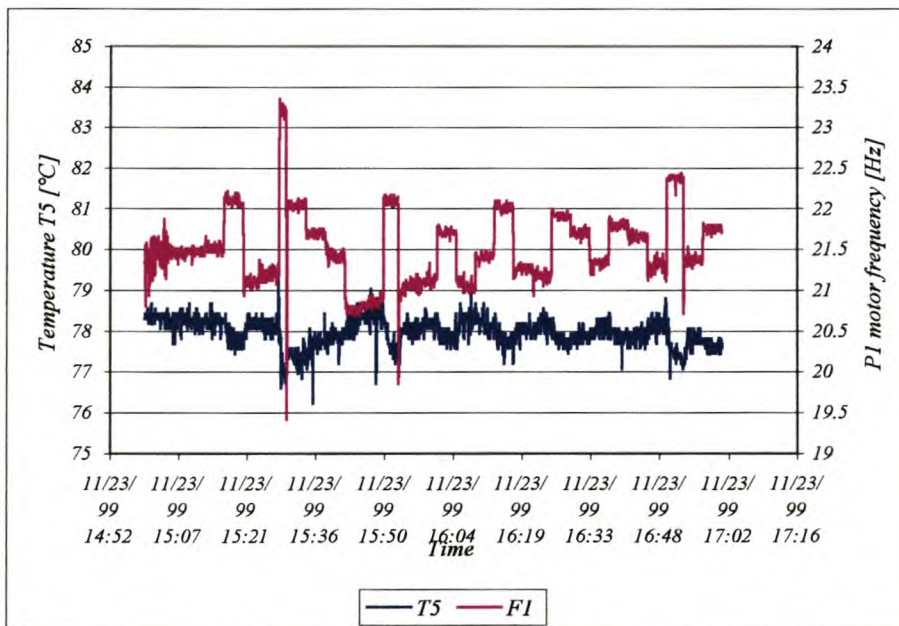


Figure 6-14 - Neurocontroller learning to control T5 at a set point of T5 = 78.3 °C.

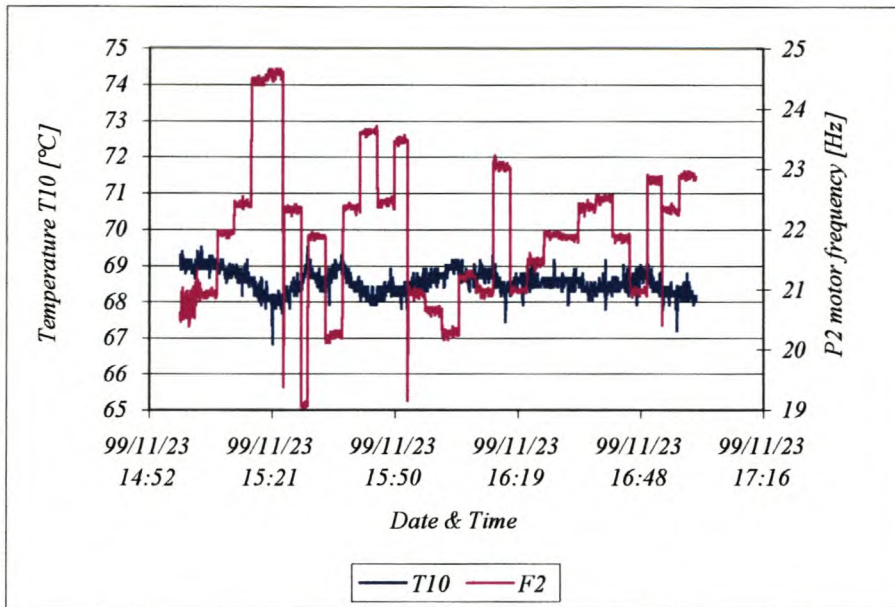


Figure 6-15 - Neurocontroller learning to control T10 at a set point of T10 = 69.0°C.

6.6 DISCUSSION OF RESULTS

To facilitate comparison between PI control and neurocontrol the *MEBAD* column was started up with the entire charge fed to the reboiler. After condensate had started to accumulate in the reflux drum, the middle vessel and reflux drum levels were controlled at 17 [%] and 25 [%] respectively, utilising PI control. These level percentages represent an accumulation of a small amount of product in each vessel, with the largest volume of liquid product retained in the reboiler. Once the plant steady state had been achieved for this liquid distribution in the column (constant temperature profile), an experimental run was commenced with either PI control or neurocontrol.

6.6.1 Temperature PI control for MEBAD column

The plant history for an experimental run from start-up utilising the PI control strategy proposed by Wittgens et al. (1996) is illustrated in figure 6-14 to figure 6-16. The PI controllers' gain and integral parameters were manually tuned to give the *ITAE* response to a set point change. The fast rise time, minimal overshoot and no oscillation for the *process variable* *T5* and *T10* (figure 6-16 and figure 6-17), is

characteristic of the *ITAE* response. The *manipulated variable* responses are also satisfactory as the reflux flow rate increases steadily in order to maintain the temperature set point in each column section. The middle vessel and reflux drum (figure 6-18) levels also have a satisfactory response to the final column steady state.

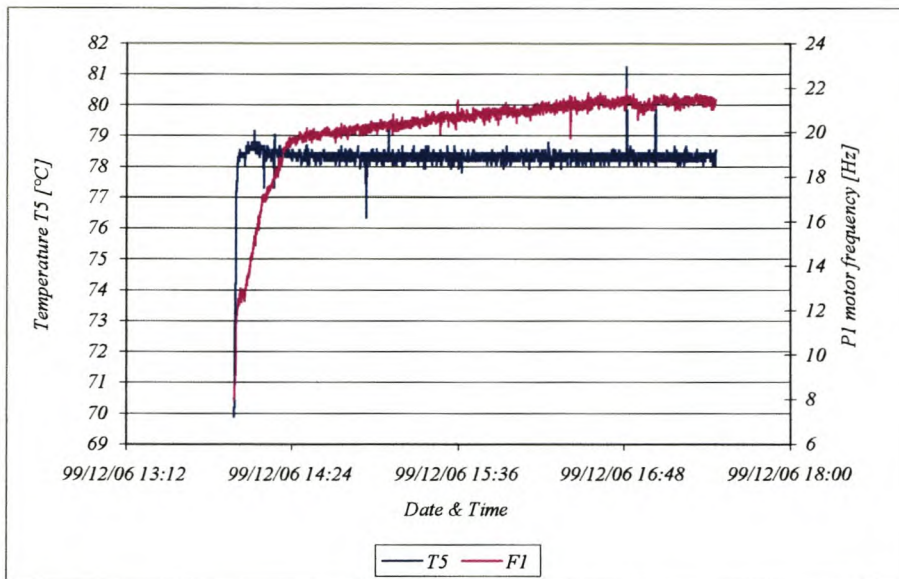


Figure 6-16 - PI Control of MEBAD column. Plant history for *process variable* T5 and pump motor frequency, F1. Set point T5 = 78.3°C.

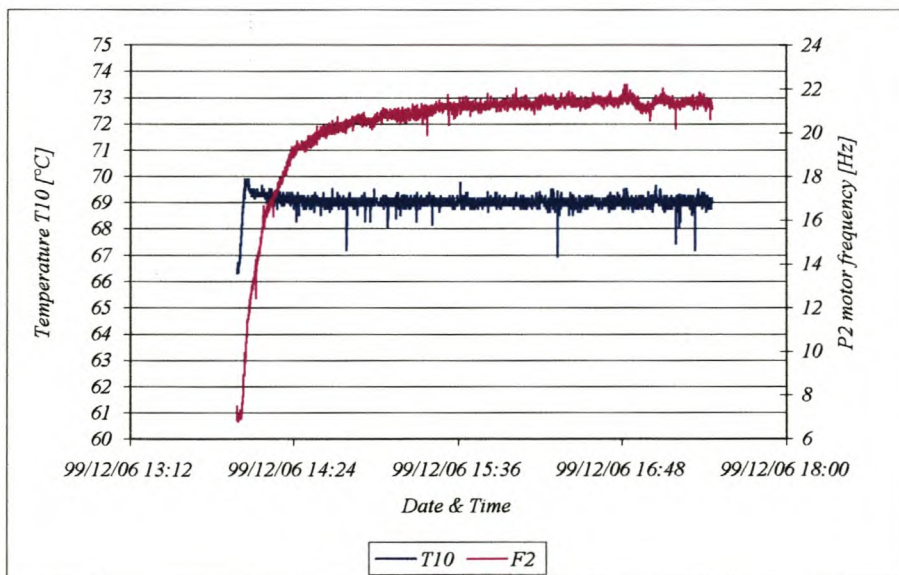


Figure 6-17 - PI Control of MEBAD column. Plant history for *process variable* T10 and pump motor frequency, F2. Set point T10 = 69.0 °C.

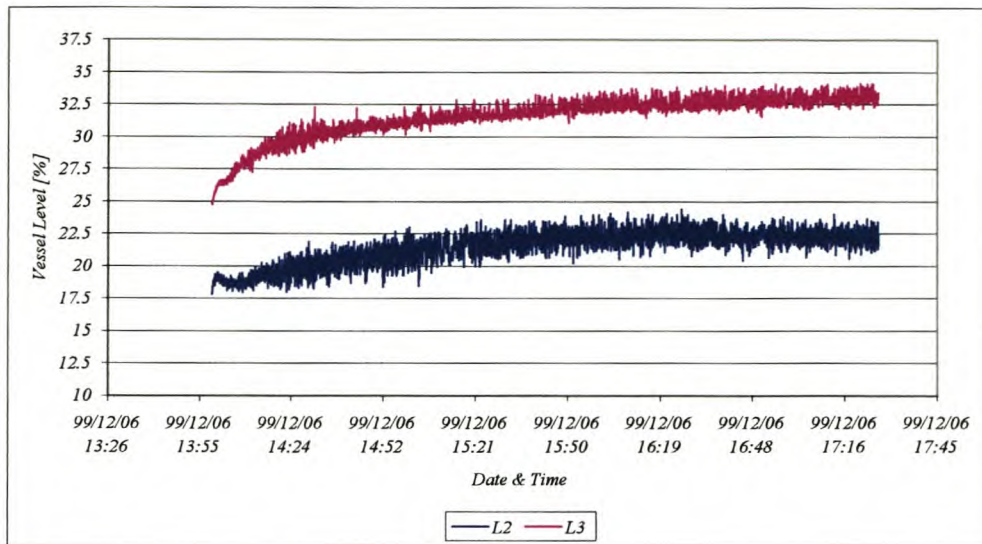


Figure 6-18 - Middle vessel and reflux drum level response utilising PI control.

6.6.2 Neurocontroller performance controlling MEBAD column

After executing the *evolutionary* algorithm for 29 generations as described in section 6.5.2.2, which constituted 145 hours of real time automated learning (experimentation), the best *neurocontroller* (with the highest *fitness*) was evaluated in the closed loop from the plant condition as described in section 6.6. The obtained responses are illustrated in figure 6-19 to figure 6-21. From figure 6-19 and figure 6-20 it is apparent that the *neurocontroller* was unable to effectively generalise to operating regions not encountered during learning. The *neurocontroller's* control *policy*, however, exhibits behaviour that reflects the correct controller gain sign. The *evolutionary* algorithm had thus discovered that as the *process variable* increases, the *manipulated variable* output must also increase (direct-acting). The *neurocontroller*, however, incorporated inappropriate controller gain magnitudes for the temperature inputs to the *network*. In order to achieve the fast rise time observed for the PI control experiments (section 6.6.1), the *neurocontroller* needs to output pump motor frequencies of approximately half that indicated in figure 6-19 and figure 6-20. As the temperature in each column section increased, the *neurocontroller* also displayed a low sensitivity to the column *process variable* magnitudes. The developed control *policy* thus results in product accumulation in the vessels that is too slow (figure 6-21), requiring the utilisation of a far greater amount of energy input than that required with PI control.

Effective control *policy* development (learning) did thus not take place. A number of possible explanations may be postulated for this occurrence.

- The nature of the process could be such, that 29 generations is an inadequate number of generations for developing an effective control *policy*. The learning rate may simply be slow, requiring further experimentation to discover an effective controller. As the current experimental data represents 145 hours of experimentation, further experimentation (even though the outcome may be successful) makes the application of *SANE* impractical, should the initialisation be from a initial random state. Instead of starting real system learning with a randomly generated *population*, some pretraining of the *population* may be possible with approximate simulation models of the system. This may significantly reduce the required practical experimentation.
- As discussed in section 2.6, non-*Markov environments* present a complex learning challenge, as all the states of the dynamic system are not available for learning. Should it be impossible for an algorithm to create an internal representation of such hidden states, the resulting behavioural *policy* may be far from optimal. For the temperature control of the *MEBAD* column, temperature readings are used to infer the composition (the true state variables) in each column section. Should a multicomponent (i.e. ternary) system require distillation, temperature measurement alone is not sufficient to infer the composition. A pseudo-binary separation results in each column section of the *MEBAD* configuration as the batch progresses, provided that a sufficient number of theoretical trays are present to afford the separation in each column. Temperature readings alone may thus, at some stage in the distillation, be effectively used to infer composition. The learning algorithm may thus only be required to deal with a partial non-*Markov* problem during start-up. In the experimental setup described in section 6.5, the column sections have an insufficient HETP for complete separation of the methanol, ethanol and n-propanol into pure components. The accurate determination of composition in the column sections is thus problematic, making internal state representation in the neural *network's* structure more difficult.

- The presence of sensor noise in the experimental setup may also have contributed significantly to lowering the learning rate. Sensor noise contributes to introducing inaccuracies into the *evaluations* of the *neurocontrollers* in the *population*. This results in inconsistent application of the genetic *operators* (such as *crossover*), as the *evaluations* may not reflect the true value of a *neurocontroller* at performing the task. Although *SANE* has been found to be robust in the presence of sensor noise (Appendix A), the slower learning rate, due to learning in the presence of sensor noise, is extremely detrimental should experimentation be costly in terms of time and finance.
- Poor *generalisation* to unencountered operating regions during learning, may also be a result of not exposing the learning *agents* to a large enough portion of the state space to affect real learning. Although *reinforcement learning* methodologies have proven highly capable of effective *generalisation* (despite learning in a small region of the state space), the available information regarding the *environment* needs to contain pertinent knowledge that may be exploited by the reinforcement algorithm in developing a control *policy*. The learning temperature bandwidth of approximately 2 [°C] (as implemented) around each set point, may not contain sufficient *environment* information for control *policy* development.
- Batch distillation remains a dynamic process throughout the batch operation. The column section temperature profiles and vessel levels are constantly changing along a trajectory dictated by the reboil and reflux rates. Although *reinforcement learning* techniques may be expected to deal with slowly changing (non-stationary) process conditions, more system information may need to be provided to the *neurocontroller* to accurately gauge the current state of the process. The temperature reading from each column section (should it be regulated at a constant temperature) gives no indication of how the batch is progressing. The *neurocontroller* may thus only increase or decrease the reflux rate to a given column section based on an positive or negative deviation from the set point. Level and other temperature sensor readings may provide more pertinent information in dealing with the non-stationary nature of batch distillation.

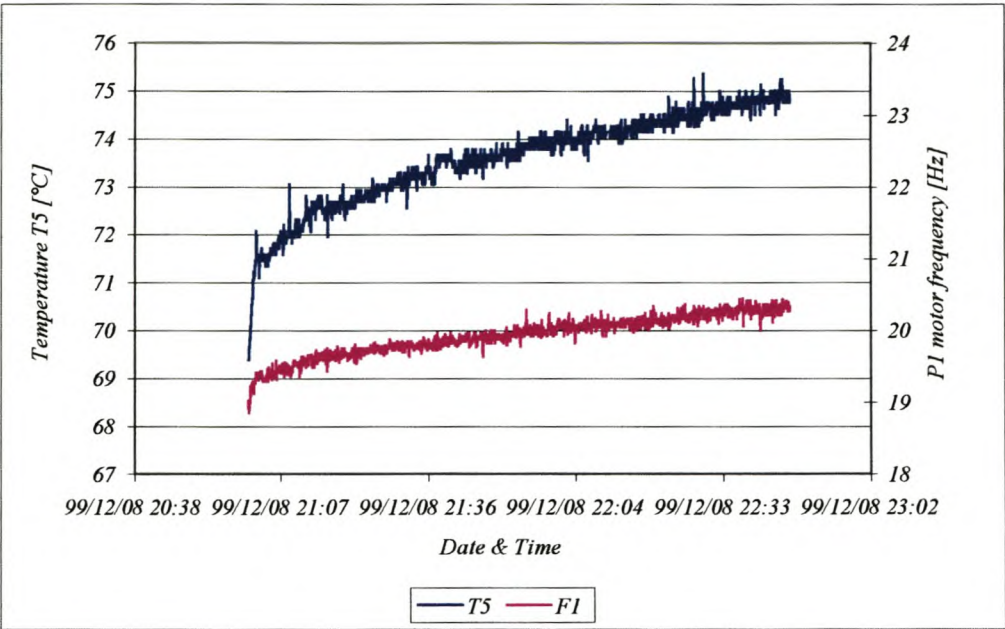


Figure 6-19 - Neurocontrol of MEBAD column. Plant history for process variable T5 and pump motor frequency, F1. Set point T5 = 78.3 °C.

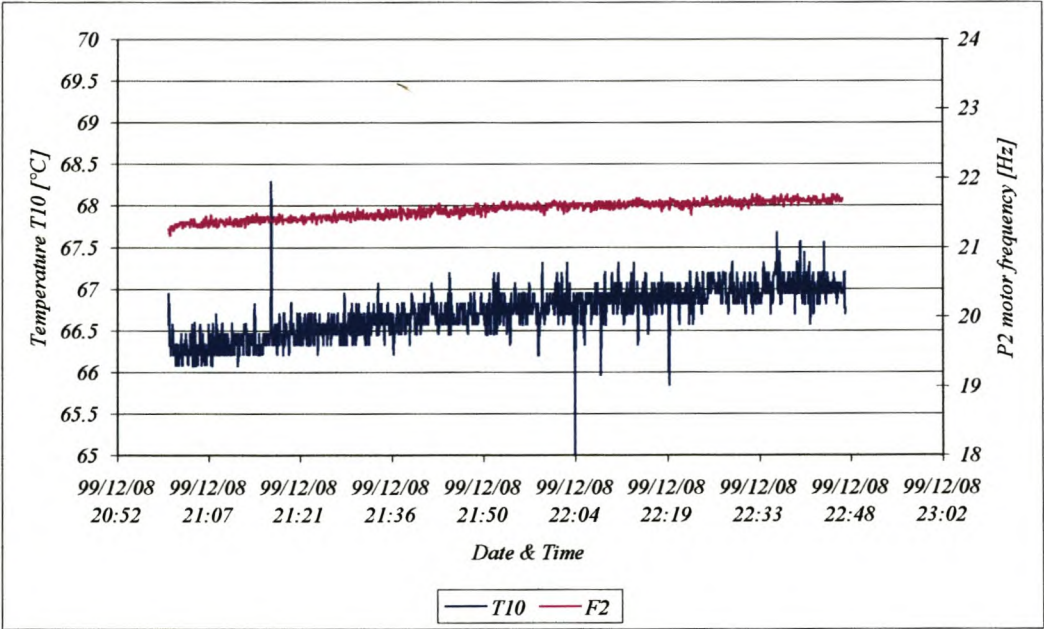


Figure 6-20 - Neurocontrol of MEBAD column. Plant history for process variable T10 and pump motor frequency, F2. Set point T10 = 69.0°C.

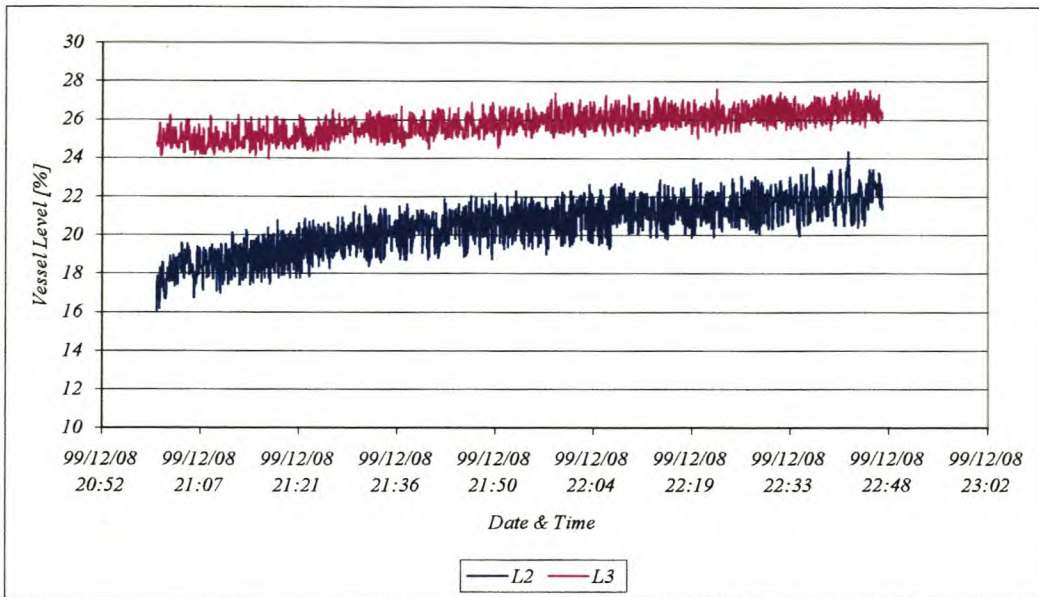


Figure 6-21 - Neurocontroller middle vessel and reflux drum levels.

Figure 6-22 provides a means of determining that the *evolutionary* process was indeed being directed to reducing the error from the desired state. Figure 6-20 illustrates the change in the error for the least fit *individual*, the *population* average and the most fit *individual*. As indicated the required controller gain sign was correctly determined in the *evolutionary* search. From figure 6-20 it is evident that the mean absolute error for the entire *population* (average) is significantly reduced as the generations progressed - the *neurocontroller population* has thus improved as a whole. It is also clear from examining figures 6-16 & 6-17 and figure 6-19 & 6-20, that the *evolutionary* algorithm had searched for an effective control *policy* within the correct range of reflux rates. The final steady state reflux rates as seen for the PI control experiments, is within ± 1 Hz of the reflux rate selected by the *neurocontroller*. Starting from a random *neurocontroller population* that could generate any reflux rate from 7 - 35 [Hz], learning proved to be fairly robust in that the appropriate region between approximately 17 - 23 Hz was being explored as viable outputs.

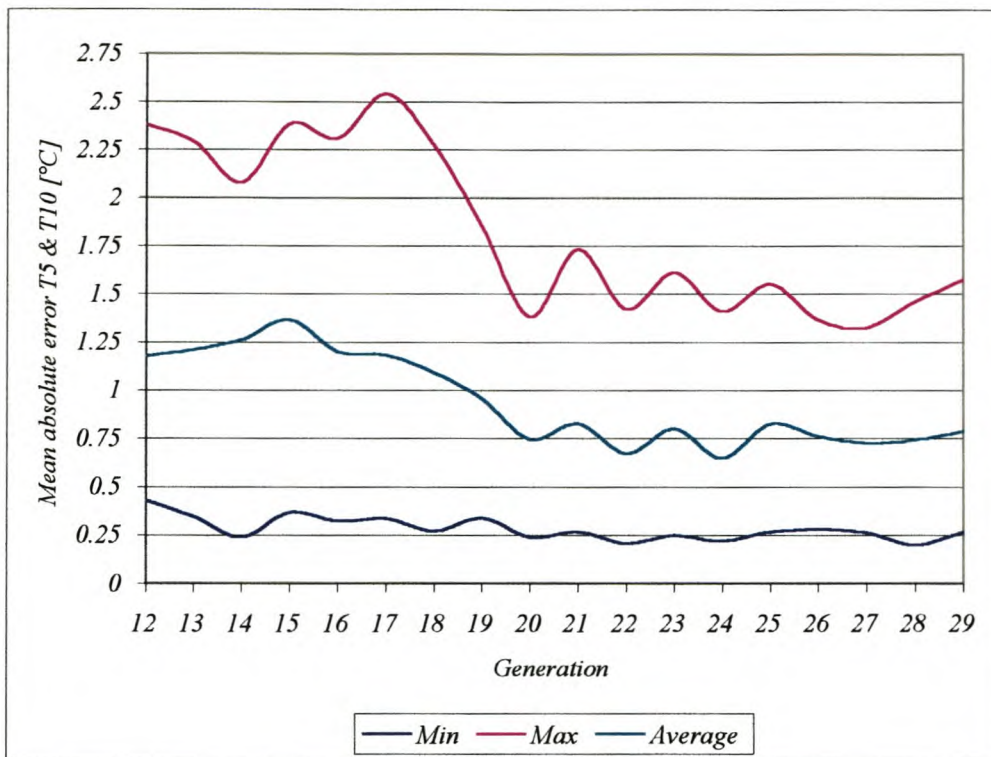


Figure 6-22 - Improvement in the sum of the absolute error from set point for process variables T5 and T10 as the evolution progressed. The population statistics include the minimum error (best controller), the average error for the population & the maximum error (worst controller).

6.7 CONCLUDING REMARKS FOR CHAPTER 6

Neurocontroller development for the real world *MEBAD* column proved less successful than for the simulation studies. Even though the differences between the *evaluation* stage in the simulation studies and that used for the batch column were minimal, learning an effective control *policy* remained elusive after 145 [h] of automated experimentation. The poor *generalisation* obtained for the developed *neurocontroller* may be attributed to the disruptive effects of the non-*Markov* and non-stationary nature of the process, sensor noise and exposure to a too small region of the state space. After tuning the PI control parameters to conform to the *ITAE* response criteria, the PI temperature control scheme proposed by Wittgens et al. (1996) provided robust control for the *MEBAD* system configuration.

6.8 SYMBOLS FOR CHAPTER 6

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
dP	Differential pressure	[Pa]
F	Pump motor frequency	[Hz]
H	Vessel hold-up	-
L	Vessel level	[%]
t	Time	-
T	Temperature	[°C]
x	Component composition	-
z	Desired component composition	-

Greek symbols

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
α	Relative volatility	-

Subscript

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
i	i'th number	-

7 CONCLUSIONS AND RECOMMENDATION FOR FUTURE WORK

7.1 CONCLUSIONS

A summary of the conclusions that may be drawn from this research are listed below -

- An *evolutionary reinforcement learning* algorithm, like *SANE*, is capable of developing both robust and high performance nonlinear controllers (*neurocontrollers*) for controlling highly nonlinear plants. The algorithm allows for the development of *neurocontrollers* that exhibit a desired closed loop response, such as the *ITAE* criteria. During simulation *evaluations*, the *SANE* learning algorithm was found to demonstrate robust learning capabilities in the presence of significant sensor noise. *SANE* may thus provide robust *credit assignment* to evaluated *neurocontrollers* in non-ideal learning *environments*.
- The developed *neurocontrollers* for the simulation studies are able to generalise behaviour learned in a small region of the state space to a wide operating range. This *generalisation* behaviour is extended to include the effective rejection of large process disturbances and sensor noise. Also, in the presence of process model uncertainties, the developed *neurocontrollers* demonstrated robust performance.
- In comparison to other advanced control techniques, such as model predictive control (*MPC*), the *neurocontrollers* developed with *SANE* proved less sensitive to model parameter uncertainties. No *model mismatch compensation* was required for the developed *neurocontrollers*, while *MPC* applications frequently incorporate the use of *model mismatch compensation* to eliminate steady state offset. The developed *neurocontrollers* are considered to exhibit greater robustness than *MPC* implementations in the presence of uncertainty, thus allowing for greater autonomy (without human interference) during process operation. Performance *evaluations* between *SANE neurocontrollers* and *MPC* controllers

were largely similar, but also dependent on the selected desired closed loop response.

- *SANE* also afforded the ability to integrate the optimum process design and process control development functions. *SANE* is able to search the *environment* for the point of maximum economic return and simultaneously develop a robust *neurocontroller* for this task. Economic considerations may be included into the *fitness* function utilised by *SANE* for *neurocontroller* development.
- *SANE* demonstrate limited success in learning an effective control *policy* for the *MEBAD* distillation column configuration. The poor *generalisation* is attributed to the non-*Markov* and non-stationary nature of the process, disruptive sensor noise in the *evaluation* of each *neurocontroller* (thus complicating *credit assignment*), and the possibility that *evaluations* were conducted in a too small region of the state space to allow for effective learning to take place. Nevertheless, the *evolutionary* algorithm improved the average *fitness* of the *population* in performing the task and had concentrated the search on the relevant area of the search space. A randomly generated initial *neuron* and *network population* results in costly experimentation in terms of time and finance. The initial *population* should be supplemented with additional pre-programmed knowledge, that may be obtained from approximate plant models.

7.2 RECOMMENDATION FOR FUTURE WORK

The following listing described possible directions for further research -

- Online learning for real world chemical processes is costly, both from a time and financial perspective. The highest possible learning rate must be obtained for the considered system. The learning rate is directly proportionate to the learning algorithm's methodology. Improvements to the *SANE* algorithm may be implemented which will increase the learning rate, thereby reducing the amount of experimentation required to find an optimal control *policy*. *SANE's* methodology for maintaining knowledge of effective *neuron* combinations via the blueprint

network population, introduces the destructive impact of the structural/functional mapping problem to the search (section 3.2). A different method for maintaining knowledge of effective *neuron* collections may avoid this complication. In addition, hybridisation of the *evolutionary* algorithm, to allow for local learning and the incorporation of adaptive algorithm techniques could speed the evolution considerably.

- For real world learning it is more practical to start the learning process from a position of knowledge. Starting the learning process from a randomly generated initial *population* requires costly experimentation. It is feasible to first pretrain the *population* on an approximate simulation model or on input-output data generated by an existing control structure (i.e., PI controllers). The online *evolutionary* process will thus only need to correct for the errors between the simulation model and the actual plant.
- Another alternative to increasing the learning rate is to combine classical control structures (i.e., PID control) and a *neurocontroller* structure to facilitate the execution of a control *policy*. Classical control elements, such as integral action, are effective tools for eliminating *state*. The incorporation of integral action into the neurocontrol structure may significantly reduce the number of generations required to obtain an effective *neurocontroller*.
- It is also feasible to include existing knowledge regarding the dynamics or controllability of a process, as inputs to the neural *network*. Should it be known, for example, that two *process variables* are closely correlated it may be possible to remove one of the *process variables* as input to the *neurocontroller*. This not only reduces the number of inputs to the neural *network* thus simplifying learning, but frees the *neurocontroller* from having to discover this correlation during learning.
- Conduct a more comprehensive analysis to demonstrate the ability of *SANE* to integrate the process design and process control development functions. Real economic data may be included into the *fitness* function of the *evolutionary*

algorithm, to allow *SANE* to find the economically optimal operating point and develop a *neurocontroller* accordingly.

- This study has mainly focused on learning to optimally control single chemical unit operations. The potential of *SANE* to develop a plant-wide control strategy for highly nonlinear plants may be investigated by simulating an entire plant in a dynamic *environment* such as HYSYS. This approach may have far reaching benefits in that the plant-wide approach will reduce controller interaction by incorporating implicit controller decoupling into the control structure.
- The *SANE evolutionary* algorithm is a robust and efficient genetic search tool, which should allow for the algorithm to also be utilised in a supervised learning capacity. The algorithm's ability to maintain *diversity* in the genetic *population*, makes it attractive for developing chemical process models from complex process data sets.
- Possible industrial implementations may be considered. It may be feasible (should an accurate model of the process exist) to develop *neurocontrollers* at more complex operating regions for industrial implementation, which may provide greater economic benefit utilising the existing plant equipment.

8 REFERENCES

- Agrawal, P., Lee, C., Lim, H.C., & Ramkrishna, D. (1982). Theoretical Investigations of Dynamic Behaviour of Isothermal Continuous Stirred Tank Biological Reactors. *Chemical Engineering Science*, 37(3), 453.
- Barolo, M., Guarise, G.B., Ribon, N., Rienzi, S., & Trotta, A. (1996). Some issues in the design and operation of batch distillation columnn with a middle vessel. *Computers & Chemical Engineering*, 20 (Supl.), S37 - S42.
- Barolo, M., & Botteon, F. (1997). Simple Method for Obtaining Pure Products by Batch Distillation. *AIChE Journal*, 43 (10), 2601 - 2604.
- Barolo, M., Guarise, G.B., Rienzi, S.A. & Trotta, A. (1998). Understanding the Dynamics of a Batch Distillation Column with a Middle Vessel. *Computers and Chemical Engineering*, 22 (Supl.), S37 - S44.
- Bequette, B.W. (1991). Nonlinear Control of Chemical Process: A Review. *Industrial & Engineering Chemistry Research*, 30, 1391 - 1413.
- Bersini, H., & Gorrini, V. (1997). A Simplification of the Backpropagation-Through-Time Algorithm for Optimal Neurocontrol. *IEEE Transactions on Neural Networks*, 8(2), 437 - 441.
- Bregel, D.D., & Seider, W.D. (1989). Multistep Nonlinear Predictive Controller. *Industrial &Engineering Chemistry Research*, 28, 1821-1822.
- Bregel, D.D., & Seider, W.D. (1992). Coordinated Design and Control Optimization of Nonlinear Processes. *Chemical Engineering Science*, 16(9), 861-886.
- Collins, R.J., & Jefferson, D.R. (1991). Selection in masively parallel *genetic algorithms*. In: Belew, R. & Booker, L. (Eds.), *Proceedings of the Fourth*

International Conference on Genetic Algorithms, 249 - 256, San Mateo, CA. Morgan Kaufmann.

Deb, K., & Goldberg, D.E. (1989). An investigation of niche and species formation in genetic function optimization. In: Schaffer, J.D. (Ed.), *Proceedings of the Third International Conference of Genetic Algorithms*, 42 - 50, San Mateo, CA. Morgan Kaufmann.

Davidor, Y. (1991). A Naturally Occuring Niche & Species Phenomenon: The Model and First Results. In: Belew, K., & Booker, L.B. (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, 257 - 263, San Mateo, CA. Morgan Kaufmann.

Davidyan, A.G., & Kiva, V.N. (1991). Separation of Binary Mixtures in Two-Section Column with Central Reservoir. *Theoretical Foundations of Chemical Engineering*, 25(4), 467-475.

Davidyan, A.G., & Kiva, V.N. (1994). Batch Distillation in a Column with a Middle Vessel. *Chemical Engineering Science*, 49(18), 3033-3051.

Eaton, J.W., & Rawlings, J.B. (1992). Model-Predictive Control of Chemical Processes. *Chemical Engineering Science*, 47(4), 705 - 720.

Economou, C.G., & Morari, M. (1986). Internal Model Control. 5. Extension to Nonlinear Systems. *Industrial & Engineering Process Des. Dev.*, 25, 403 - 411.

Engell, S., & Klatt, K.U. (1993). Gain Scheduling of a Non-minimum Phase CSTR. In: Niewenhuis, J.W., Praagman, C., Trentelman, H.L. (Eds.), *Proceedings of the 2nd European Control Conference*, 2323 - 2328, Groningen, IEEE Control Systems Society.

Farschman, C.A., & Diwekar, U. (1998). Dual Composition Control in a Novel Batch Distillation Column. *Industrial & Engineering Chemistry Research*, 37, 89 - 96.

Gattu, G., & Zafirou, E. (1992). Nonlinear Quadratic Dynamic Matrix Control with State Estimation. *Industrial & Engineering Chemistry Research*, 31, 1096 - 1104.

Greffentette, J.J., Ramsey, C.L., & Schultz A.C. (1990). Learning Sequential Decision Rules Using Simulation Models and Competition. *Machine Learning*, 5, 355-381.

Gupta, M.M., & Rao, D.H. (1994). Neuro-Control Systems: A Tutorial. In Gupta, M.M., & Rao, D.H. (Eds.), *NeuroControl Systems: Theory and Application*, 1-43. IEEE Press.

Harris, K.R., & Palazoglu, A. (1998). Studies on the analysis of nonlinear processes via functional expansions - III: controller design. *Chemical Engineering Science*, 53(23), 4005 - 4022.

Hasebe, S., Abdul Aziz, B.B., Hashimoto, I., & Watanabe, T. (1992). Optimal Design and Operation of Complex Batch Distillation Column. In: *Preprints of the IFAC Workshop on Interaction between Process Design and Process Control*, 177 - 182, London.

Hasebe, S., Kurooka, T., & Hashimoto, I. (1995). Comparison of the Separation Performances of a Multi-Effect Batch Distillation System and a Continuous Distillation System. In: *Proceedings IFAC-symposium DYCORS'95*, 249 - 254, Denmark.

Hasebe, S., Kurooka, T., Abdul Aziz, B.B., Hashimoto, I., & Watanabe, T. (1996). Simultaneous Separation of Light and Heavy Impurities by a Complex Batch Distillation Column. *Journal of Chemical Engineering of Japan*, 29(6), 1000 - 1006.

Hasebe, S., Noda, M., & Hashimoto, I. (1997). Optimal operation policy for multi-effect batch distillation system. *Computers and Chemical Engineering*, 21, S1221-1226.

Hecht-Nielsen, R. (1988). Neurocomputing: picking the human brain. *IEEE Spectrum*, 25(3), 36-41.

Henson, M.A., & Seborg, D.E. (1992). Nonlinear Control Strategies for Continuous Fermenters. *Chemical Engineering Science*, 47(4), 821-835.

Horn, J., Goldberg, D.E., & Deb, K. (1994). Implicit niching in the learning classifier system: Nature's way. *Evolutionary Computation*, 2(1), 37-66.

Horn, J., & Goldberg, D.E. (1996). Natural Niching for Evolving Cooperative Classifiers. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., & Riolo, R.L. (Eds.), *Genetic Programming, Proceedings of the First Annual Conference 1996*, 553-564, Cambridge (MA), The MIT Press.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, 2, 359-366.

Hunt, K.J., Sbarbaro, D., Zbikowski, R. & Gawthrop, P.J. (1992). Neural Networks for Control Systems - A Survey. *Automatica*, 28(6), 1083 - 1112.

Husbands, P., & Mill, F. (1991). Simulated Co-Evolution as the Mechanism for Emergent Planning and Scheduling. In: Belew, R. & Booker, L. (Eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms*, 264 - 270, San Mateo, CA. Morgan Kaufmann.

Kaelbling, L.P., Littman, M.L., & Moore, A.W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4, 237-285.

Kim, S., Lee, M., Park, S., Lee, S. & Park, C.H. (1997). A Neural Linearizing Control Scheme for Nonlinear Chemical Processes. *Computers and Chemical Engineering*, 21(2), 187 - 200.

Li, W.C., & Biegler, L.T. (1988). Process Control Strategies for Constrained Nonlinear Systems. *Industrial & Engineering Chemistry Research*, 27, 1421 - 1433.

Mahfoud, S.W. (1992). Crowding and Preselection Revisited. In: Manner, R., & Manderick, B. (Eds), *Parallel Problem Solving from Nature*, 2, 27 -36, Amsterdam, Elsevier Science Publishers.

Matsuura, T., & Kato, M. (1967). Concentration stability of the isothermal reactor. *Chemical Engineering Science*, 22, 171 - 184.

Minsky, M.L. (1961). Steps toward artificial intelligence. *Proceedings of the Institute of Radio Engineers*, 49, 8 - 30.

Moriarty, D.E, & Miikkulainen, R. (1996a). Efficient Reinforcement Learning through Symbiotic Evolution. *Machine Learning*, 22, 11-32.

Moriarty, D.E., & Mikkulainen, R. (1996b). Hierarchical Evolution of Neural Networks. *Technical Report AI96-242*, Department of Computer Science, The University of Texas at Austin.

Moriarty, D.E. (1997). Symbiotic Evolution of Neural Networks in Sequential Decision Tasks. Ph.D. Thesis. University of Texas (Austin).

Moriarty, D.E., & Miikkulainen, R. (1998). Forming Neural Networks through Efficient and Adaptive Coevolution. *Evolutionary Computation*, 5(4), 1-28 .

Mujtaba, I.M., & Macchietto, S. (1994). Optimal Operation of Multicomponent Batch Distillation - A Comparative Study Using Conventional and Unconventional Columns. In: Preprints of ADCHEM'94, Kyoto, May 1994, 415-420.

Mujtaba, I.M. (1992). Use of Continuous Distillation Columns for Batch Separation. *Computers & Chemical Engineering*, 16(S), S273 - S280.

Patwardhan, S.C., & Madhavan, K.P. (1993). Nonlinear Model Predictive Control Using Second-Order Model Approximation. *Industrial & Engineering Chemistry Research*, 32, 334 - 344.

- Paredis, J. (1997). Coevolutionary Process Control. In: Smith, G.D. (Ed.). *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA97)*. Springer, Vienna.
- Potter, M.A., & De Jong, K.A. (1995). Evolving Neural Networks with Collaborative Species. In: Ören, T.I., & Birta, L.G. (Eds.), *Proceedings of the 1995 Summer Computer Simulation Conference*, Ottawa, Canada, 340 - 345.
- Psichogios, D.C., & Ungar, L.H. (1991). Direct and Indirect Model based Control Using Artificial Neural Networks. *Industrial & Engineering Chemistry Research*, 30, 2564 - 2573.
- Puskorius, G.V., & Feldkamp, L.A. (1994). Neurocontrol of Nonlinear Dynamic Systems with Kalman Filter Trained Recurrent Networks. *IEEE Transactions on Neural Networks*, 5(2), 279 - 297.
- Safrit, B.T., & Westerberg, A.W. (1995). Extending Continuous Conventional and Extractive Distillation Feasibility Insights to Batch Distillation. *Industrial & Engineering Chemistry Research*, 34, 3257 - 3264.
- Safrit, B.T., & Westerberg, A.W. (1997). Improved Operational Policies for Batch Extractive Distillation Columns. *Industrial & Engineering Chemistry Research*, 36, 436 - 443.
- Seider, W.D., Brengel, D.D., & Provost, A.M. (1990). Nonlinear Analysis in Process Design. Why OverDesign To Avoid Complex Nonlinearities? *Industrial & Engineering Chemistry Research*, 29, 805 - 818.
- Skogestad, S., Wittgens, B., Sørensen, E., & Litto, R. (1995). Multivessel Batch Distillation. *AIChE Annual Meeting*, Miami Beach, Paper 184i.
- Skogestad, S., Wittgens, B., Sørensen, E., & Litto, R. (1997). Multivessel Batch Distillation. *AIChE Journal*, 43(4), 971 - 978.

Smith, R.E., Forrest, S., & Perelson A.S. (1993). Searching for diverse, cooperative *populations* with genetic algorithms. *Evolutionary Computation*, 1(2), 127-149.

Stephanopoulos, G., & Han, C. (1996). Intelligent Systems in Process Engineering: A Review. *Computers and Chemical Engineering*, 20(6/7), 743 - 791.

Sutton, R.S., & Barto, A.G. (1998). Reinforcement Learning - An Introduction. A Bradford Book. Cambridge, Massachusetts.

Tanese, R. (1989). Distributed genetic algorithms. In: Schaffer, J.D. (Ed.), *Proceedings of the Third International Conference of Genetic Algorithms*, 434 - 439, San Mateo, CA. Morgan Kaufmann.

Ungar, L.H. (1991). A Bioreactor Benchmark for Adaptive Network-based Process Control. In: Miller, W.T., Sutton, R.S., & Werbos P.J. (Eds.), *Neural Networks for Control*, 387 - 402, Cambridge (Mass), The MIT Press.

Weeks, E.R., & Burgess, J.M. (1997). Evolving artificial neural networks to control chaotic systems. *Physical Review E*, 56(2), 1531-1540.

Whitehead, S.D., & Lin, L (1995). Reinforcement learning of non-*Markov* decision processes. *Artificial Intelligence*, 73, 271-306.

Whitley, D., & Starkweather, T. (1990). Optimizing Small Neural Networks using a Distributed Genetic Algorithm. In: *Proceedings of 1990 International Joint Conference on Neural Networks*, 206 - 209, Hillsdale (NJ), Lawrence Erlbaum.

Whitley, D., Starkweather, T., & Bogart, C. (1990). Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing*, 14, 347-361.

Whitley, D., Dominic, S., Das, R., & Anderson, C.W. (1993). Genetic Reinforcement Learning for Neurocontrol Problems. *Machine Learning*, 13, 259-284.

Wittgens, B., Litto, R., Sørensen, E., & Skogestad, S. (1996). Total Reflux Operation of Multivessel Batch Distillation. *Computers and Chemical Engineering*, 20 (Suppl.), S1041 - S1046.

Wittgens, B., & Skogestad, S. (1997). Multivessel Batch Distillation - Experimental Verification. In: Symposium Distillation and Adsorption 97, Maastricht, Netherlands.

Wittgens, B., & Skogestad, S. (1998). Multivessel Batch Distillation - Potential Energy Savings. IFAC Symposium, Dycops '98, June 1998, Corfu (Greece).

Yao, X. (1999). Evolving Artificial Neural Networks. *Proceedings of the IEEE*, 87(9), 1423-1447.

9 APPENDIX A

A HISTORICAL MACHINE LEARNING BENCHMARK - THE INVERTED PENDULUM

OBJECTIVES OF APPENDIX A

- Discuss the inverted pendulum machine learning benchmark, historically used to evaluated *reinforcement learning* algorithms.
- Investigate the ability of *SANE* to not only exhibit high learning rates in obtaining a stable control policy, but also in obtaining a control *policy* that matches a desired closed loop response for the system.
- Evaluate linear and neurocontrol approaches for obtaining an effective controller.
- Compare the linear and neurocontrol solutions for robustness, with regard to model parameter uncertainty and disturbance rejection.
- Evaluate the learning robustness of *SANE* for use in real world applications, by investigating learning rate for non-*Markov* processes and in the presence of noise on the state variable inputs.

9.1 THE INVERTED PENDULUM PROBLEM

The pioneering early work in machine learning for control purposes, was strongly supported and shaped by a few benchmark control problems. The characteristics of the problem being solved and the techniques used to solve it interact closely; different problems call for different approaches and different control architectures and algorithms. Solving a benchmark problem is thus only useful should it be apparent as to why the problem is difficult to solve (Ungar, 1991).

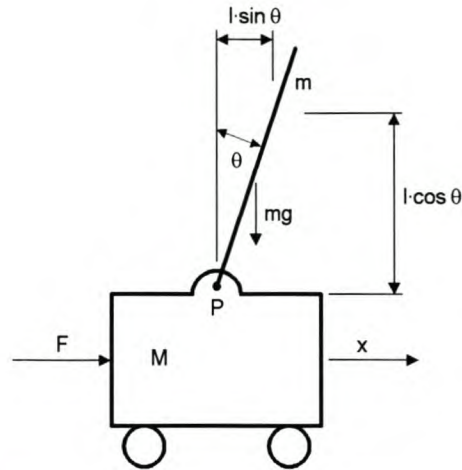


Figure 9-1 - Inverted Pendulum

An influential early model control problem is the problem of balancing a pole (inverted pendulum) on a moving cart (figure 9-1). A single pole is centred on a cart, with both pole and cart moving along a two dimensional track. An unstable open loop control problem results, as any movement of the cart on the horizontal track tends to unbalance the pole. Most frequently, the possible actions applied to the cart are discrete, that is, a positive or negative force of fixed magnitude (either -10N or $+10\text{N}$). The *agent* is thus only required to decide from which direction to apply the fixed force to the cart. The state variables of the inverted pendulum are either regarded as continuous or divided into a small set of discrete regions (direction the pole leans, rough position of the cart) for sensory purposes. Reinforcement is only awarded once the pole angle exceeds a predefined maximum deviation, or the cart exceeds the bound of the track. This problem has frequently been used as a test bed for trying and comparing new algorithms. The difficulty of the problem lies in the delayed reinforcement and inaccurate measurement of state variables. The delayed reinforcement presents a challenging *credit assignment* problem (section 2.5), as the *reward* is only allotted to the *agent* once numerous different actions have contributed to either failure or success. The pole balancing problem has been a useful benchmark in that it captures the aspects of delayed feedback (Ungar, 1991).

As a *Markov* decision process (section 2.6) the *agent* is afforded the following state information: the position of the cart (x), the velocity of the cart (\dot{x}), the angle of the

pole (θ), the angular velocity of the pole ($\dot{\theta}$). The cart and pole system may be described by the following nonlinear second-order equations of motion with the parameter formulation in table 9-1 -

$$\ddot{\theta} = \frac{M \cdot g \cdot \sin \theta - \cos \theta \cdot \left(F + m \cdot l \cdot \dot{\theta}^2 \cdot \sin \theta \right)}{\left(\frac{4}{3} \right) \cdot M \cdot l - m \cdot l \cdot \cos^2 \theta} \quad (9-1)$$

$$\ddot{x} = \frac{F + m \cdot l \cdot \left(\dot{\theta}^2 \cdot \sin \theta - \ddot{\theta} \cdot \cos \theta \right)}{M} \quad (9-2)$$

Table 9-1 - Model parameter formulation for the inverted pendulum

<i>Description</i>	<i>Formulation</i>	<i>Unit</i>
Cart position from track centre	x	$[m]$
Cart velocity	\dot{x}	$\left[\frac{m}{s} \right]$
Pole angle from point P	θ	$[^\circ]$
Pole angular velocity	$\dot{\theta}$	$\left[\frac{^\circ}{s} \right]$
Length of pole	$l=0.5$	$[m]$
Mass of pole	$m = 0.1$	$[kg]$
Cumulative mass pole & cart	$M = 1.1$	$[kg]$
Applied force	$F = 10$	$[N]$
Gravitational acceleration	$g = 9.81$	$\left[\frac{m}{s^2} \right]$

A fourth order Runge-Kutta (or equivalent method) may be utilised to solve the equations of state. Typically a discrete time step of 0.02 [s] is utilised in simulation studies. Once the pole falls beyond $\pm 12^\circ$ or the cart exceeds the 4.8 [m] track, the trial ends and a reinforcement signal is generated. Generally, performance is measured by the number of time steps for which the pole remains balanced. A control strategy was

deemed successful should the pole be balanced for 120 000 time steps (Moriarty & Mikkulainen, 1996).

Moriarty & Mikkulainen (1996) compared the performance of *SANE* with other *reinforcement learning* paradigms. These techniques included *Q-learning*, GENITOR and Adaptive Heuristic Critic (1 layer & 2 layer approach). *SANE* was found to have improved performance in learning speed and *generalisation* ability. The implementation in *SANE* involved developing a *network* with 4 input nodes (4 state variables), 8 hidden nodes with 40 connections and 2 output nodes. The output node with the highest activation determines from which direction the force is applied.

9.2 A PERFORMANCE BASED FORMULATION OF THE INVERTED PENDULUM PROBLEM

The *evaluation* of *reinforcement learning* algorithms with the above problem statement in section 9.1 for control of the inverted pendulum, has primarily focused on stabilising the open loop unstable plant with a force of fixed magnitude. Most *evaluations* of *reinforcement learning* algorithms have given no consideration to controller performance in the closed loop; that is, the controller's ability to track the plant's set point(s) with minimal error. In the proposed performance based formulation, the controller must determine both the direction and the magnitude of the force to be applied. For the inverted pendulum the controller must bring the cart to the centre (0m) of the track and maintain the pole at 0° , from any starting state and in the presence of disturbances. This problem statement results in a multi-input-single-output (MISO) control objective, with the pole angle and cart position as set points, the four state variables as *process variables* and the vector of the applied force as the *manipulated variable*.

For the development of the inverted pendulum controller, the desired closed loop response is determined by using the *ITAE* (integral time absolute error) prototype design. The *ITAE* responses minimise the integral of the time multiplied by the absolute value of the error, that is, to minimise -

$$J = \int_0^{\infty} t \cdot |e| \cdot dt \quad (9-3)$$

This controller performance criteria was applied to both the linear and *neurocontroller* development. The development of the linear controller and *neurocontroller* is discussed in section 9.3 below.

9.3 MODERN LINEAR CONTROL AND NEUROCONTROL SOLUTIONS TO THE INVERTED PENDULUM PROBLEM

9.3.1 Development of the neurocontroller

A feedforward partially connected neural *network* was developed, to have the desired *ITAE* control response in the closed loop, using the *SANE* algorithm. *SANE's* *fitness* function thus related how closely the controlled response matched the minimisation criteria in equation 9.3. The reinforcement *agent* (*neurocontroller*) learned to control the inverted pendulum using the nonlinear process model. The developed *neurocontroller* has 4 input nodes (state variables), 8 hidden nodes with 80 connections, and 4 output nodes. Two of the output nodes determines (highest activation) the direction of the applied force, while the remaining two output nodes specifies the magnitude of the force to be applied. The magnitude of the applied force is determined by multiplying the selected output node's activation by 10N. Only the pole angle and the cart position were required to conform to the *ITAE* response specification. The angular velocity and the cart velocity were thus not required to conform to a particular response, the responses for these two state variables were indirectly determined by the behaviour adopted by the *neurocontroller*.

The resulting transient response, from an initial state of $\theta = 6^\circ$ and $x = 1.2$ [m], for each state variable is illustrated in figures 9-2 to figure 9-5. The *manipulated variable* control action is illustrated in figure 9-6. The transient responses for the pole angle and the cart position match the *ITAE* specification for this system. *SANE* thus facilitated the design of a *neurocontroller* with the desired dynamic response.

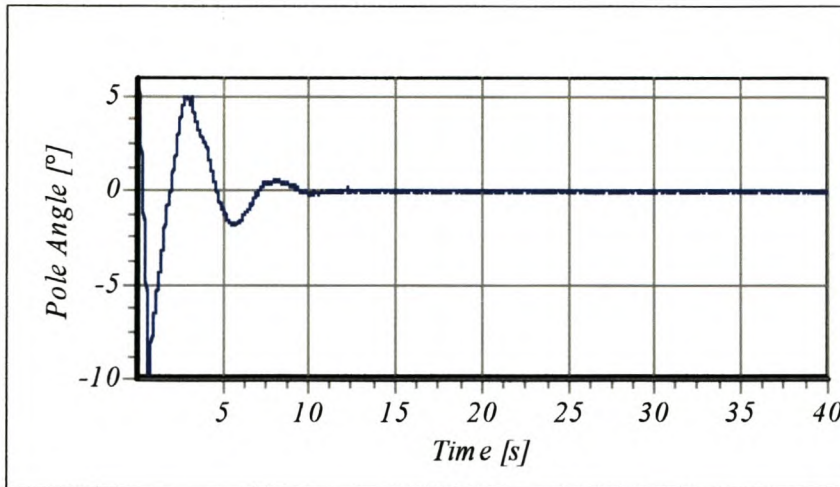


Figure 9-2 - Pole angle transient response for the neurocontroller from an initial state $\theta = 6^\circ$ and $x = 1.2$ m.

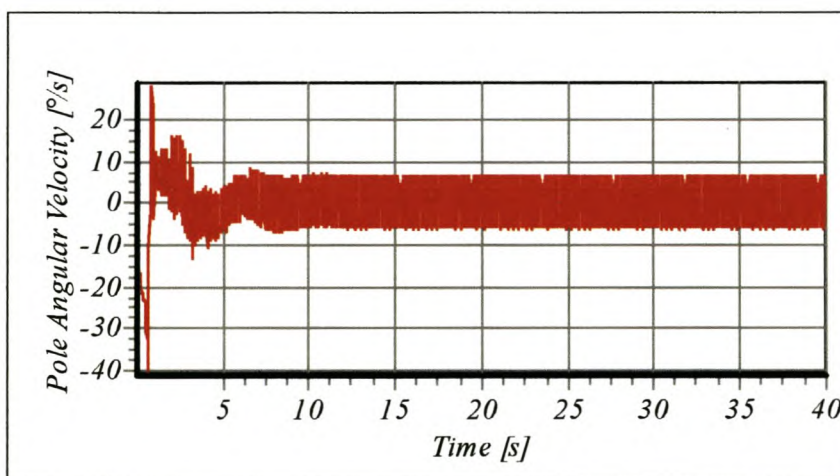


Figure 9-3 - Pole angular velocity transient response for neurocontroller from an initial state $\theta = 6^\circ$ and $x = 1.2$ m.

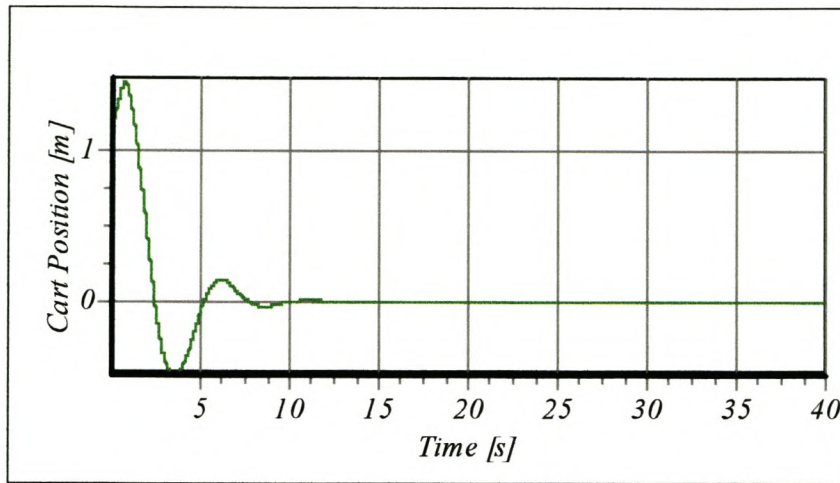


Figure 9-4 - Cart position transient response for neurocontroller from an initial state $\theta = 6^\circ$ and $x = 1.2\text{m}$.

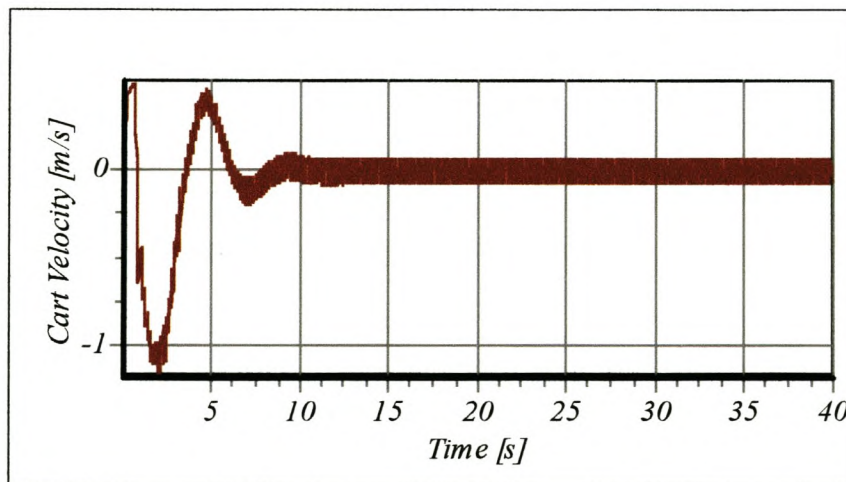


Figure 9-5 - Cart velocity transient response for neurocontroller from an initial state $\theta = 6^\circ$ and $x = 1.2\text{m}$.

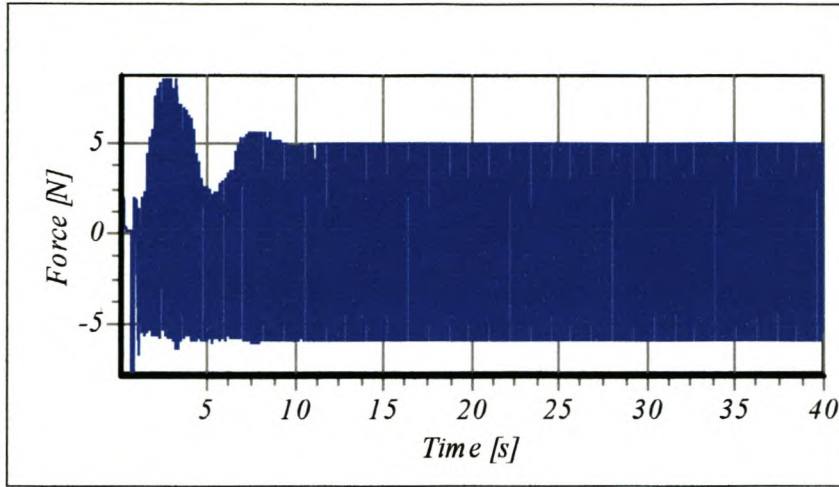


Figure 9-6 - Neurocontroller manipulated variable actions resulting in the state variable response in figure 5-2 to figure 5-6.

9.3.2 Development of the linear controller

When the pole angle is assumed small ($< 12^\circ$) the nonlinear model may be linearised by approximating $\sin(\theta)$ and $\cos(\theta)$ with θ (in radians). This allows for a linear controller's development using linear state space design techniques. The linearised plant model may consequently be expressed by the following two equations -

$$M \cdot l \cdot \ddot{\theta} = (M + m) \cdot g \cdot \theta - u \quad (9-4)$$

$$M \cdot \ddot{x} = u - m \cdot g \cdot \theta \quad (9-5)$$

For the state space design the state variables are defined as $x_1 = \theta$, $x_2 = \dot{\theta}$, $x_3 = x$, $x_4 = \dot{x}$. θ and x are the system outputs as defined in equation 9.6.

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \theta \\ x \end{bmatrix} = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} \quad (9-6)$$

with the equations of state described by equation 9-7 ($\dot{x} = A \cdot x + B \cdot u$).

$$\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \\ \ddot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{M+m}{M \cdot l} g & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{m}{M} g & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ \frac{1}{M} \end{bmatrix} \cdot u \quad (9-7)$$

The numerical values of the various variables are as presented in table 9-1. A cut-off frequency of $1.5 \frac{rad}{s}$ for the *ITAE* prototype design was selected to allow for a settling time similar to that obtained for the developed neurocontrol (figure 9-2 to figure 9-6). Using MATLAB's place function to obtain the desired closed loop poles the linear controller's control signal, u , was determined as shown in equation 9.8.

$$u = -K \cdot x = 14.745 \cdot x_1 + 1.8072 \cdot x_2 + 0.258 \cdot x_3 + 0.4644 \cdot x_4 \quad (9-8)$$

The resulting transient response, from an initial state of $\theta = 6^\circ$ and $x = 1.2$ [m], for the linear controller for each state variable is illustrated in figures 9-7.

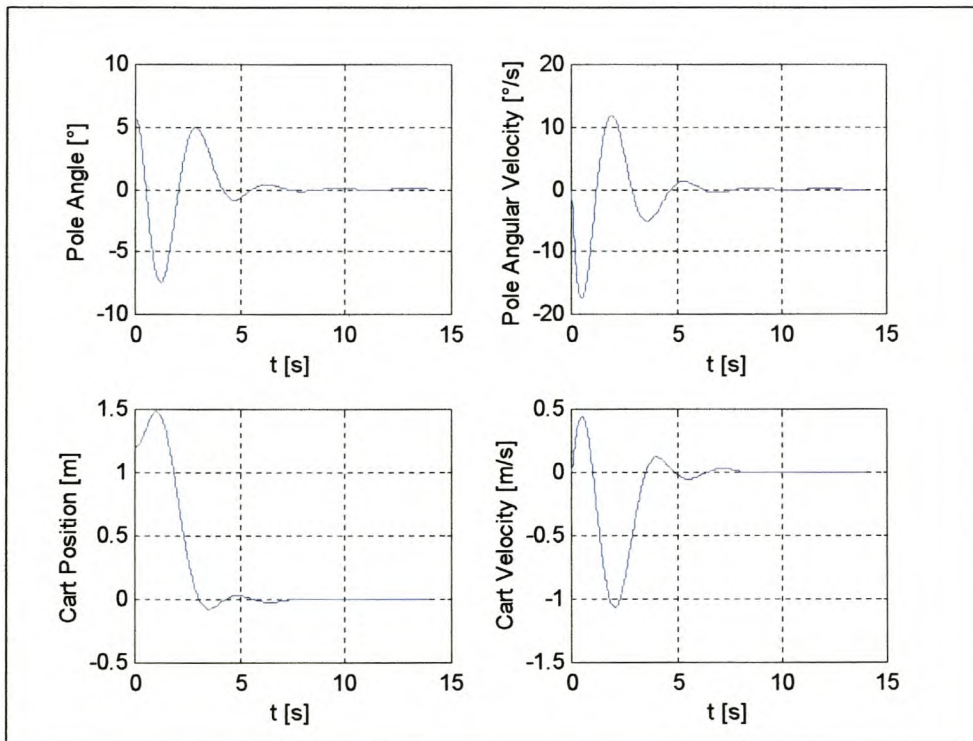


Figure 9-7 - Response of linear controller developed on the linearised model, from an initial condition of $x_1 = 0.1rad$ & $x_2 = 1.2m$.

9.4 COMPARATIVE AND SENSITIVITY ANALYSIS FOR CONTROLLER ROBUSTNESS

9.4.1 Linear controller performance in the nonlinear plant control loop

The transient responses for the *neurocontroller* and the linear controller for the pole angle and the cart position from the same initial condition, are similar as may be seen from a comparison of figure 9-1, figure 9-3 and figure 9-7. The rise time, decay ratio and the settling time are closely matched. Both controller's thus comply with the *ITAE* specification in the state variables for which the specification was required.

As the development of the linear controller was based on a simplified model for the inverted pendulum, the performance of the linear controller in the nonlinear closed loop was examined for robustness. From the same initial state condition used to illustrate the performance in the linear closed loop, the below transient responses for the linear controller were obtained in the nonlinear closed loop, as illustrated in figure 9-8 to figure 9-10. The temporal state variable responses are omitted for brevity.

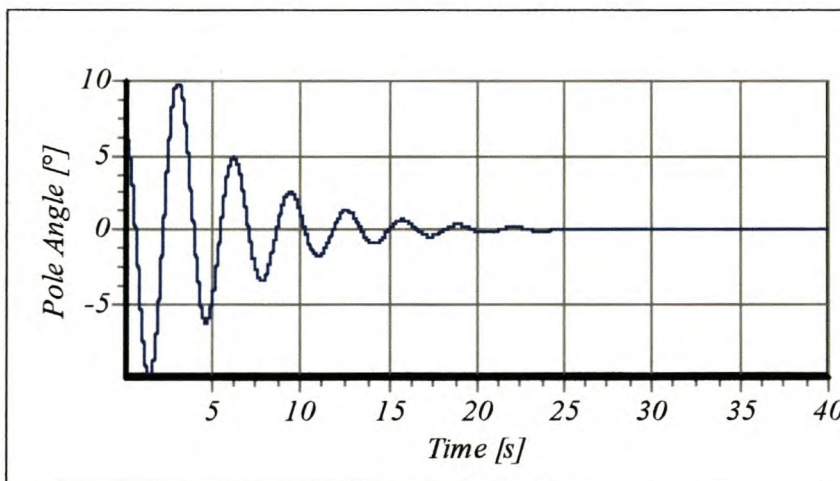


Figure 9-8 - The transient response for the pole angle in the nonlinear closed loop, from an initial condition of $\theta=6^\circ$ and $x = 1.2\text{m}$.

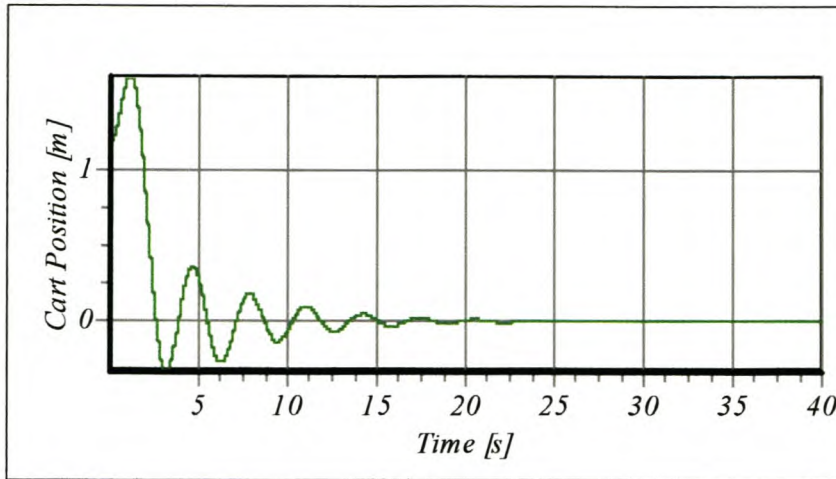


Figure 9-9 - The transient response for the cart position in the nonlinear closed loop, from an initial condition of $\theta=6^\circ$ and $x = 1.2\text{m}$.

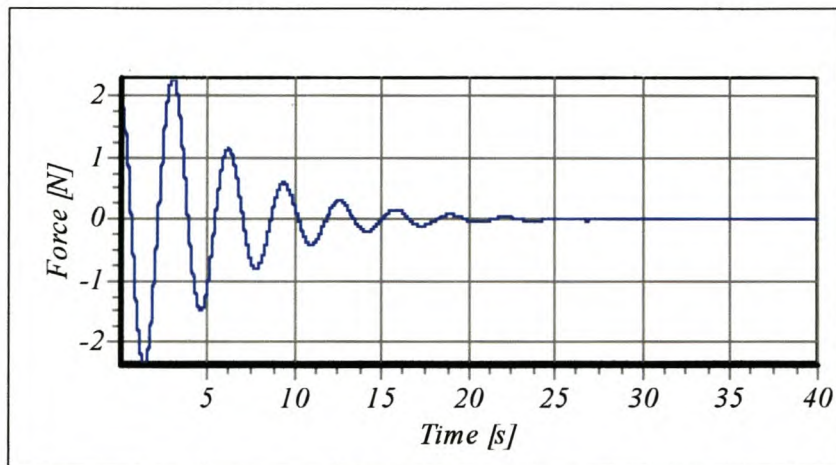


Figure 9-10 - Manipulated variable action for the linear controller in a closed loop with the nonlinear plant.

Even though the assumptions that lead to the linearisation of the nonlinear model appear justified, the response for the developed linear controller is more underdamped (oscillatory) in the nonlinear closed loop (figure 9-8 to figure 9-12), than in the linear closed loop on which it was developed (figure 9-7). The transient for the linear controller in the nonlinear *domain* has a settling time more than double that encountered for the linear plant. This is attributed to the small yet significant difference between the linearised model and the actual nonlinear model. In order to apply linear modern control techniques, linearisation of a nonlinear state space is required. Developing a controller on a linearised model thus does not guarantee

satisfactory performance in the closed loop, even though the assumptions for linearisation appear appropriate. Overcoming this linear/nonlinear model mismatch requires an iterative approach to controller design. A developed linear controller may be tested in the nonlinear system *environment*, followed by redesigned on the linear model dictated by whether the closed response in the nonlinear closed loop is satisfactory. In this case, desired closed loop poles may be selected which exhibit more damping than those initially considered. Excessive oscillatory responses in the nonlinear closed loop may be reduced in this manner. It may prove difficult to maintain acceptable settling times, as "detuning" the linear controller will typically result in longer settling times for a less oscillatory response.

Utilising nonlinear design techniques that develop controllers utilising more accurate nonlinear models may thus prove advantageous, even for system that have mild nonlinearity. *Neurocontrollers*, which are inherently nonlinear in structure, and may be designed utilising nonlinear models, may thus prove to have considerable performance advantages over linear design techniques that require simplified models in order to apply the design routines.

9.4.2 Comparative controller performance under plant/model mismatches

Where model parameter uncertainty exists or model parameters are assumed to vary unpredictably, it is prudent to test the developed controller's robustness to varying model parameters. The length of the pole for the inverted pendulum was selected as variable to perform a sensitivity analysis on the robustness of the linear and *neurocontroller*. Increasing the pole length has a significant impact on the pole angular velocity. The allowable pole angle deviation during the simulation was set to 20° for this robustness test, to allow for larger pole angle deviations as a result of the larger pole angular velocities. The response for the linear and *neurocontroller* was examined over a 40s simulation period. The controller was deemed to have failed should the 20° pole angle deviation limit or the normal boundaries of the track be exceeded. Increasing the pole length from 0.5m to 0.60m for the linear controller (18% increase in pole length) resulted in the linear controller exceeding the allowed 20° pole angle deviation in the 40s simulation. The response with a pole length of

0.59m (able to remain in the required pole angle bandwidth for 40s) is illustrated in figure 9-11 to figure 9-12 (the temporal state variable responses are omitted for brevity). From figure 9-11 and figure 9-12 it is apparent that the amplitude of each state variable response is also slowly increasing with time, indicating that the linear controller is unstable for a pole length of 0.59m.

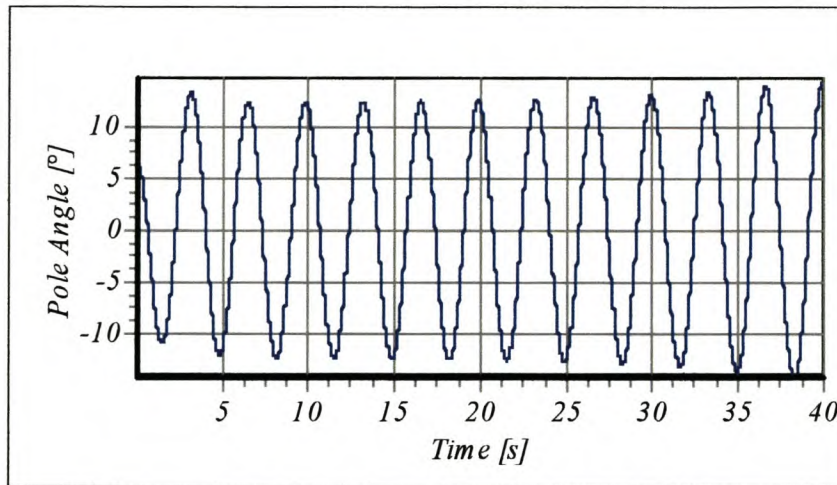


Figure 9-11 - Linear controller transient response for the pole angle with a pole length of 0.59m in the nonlinear closed loop.

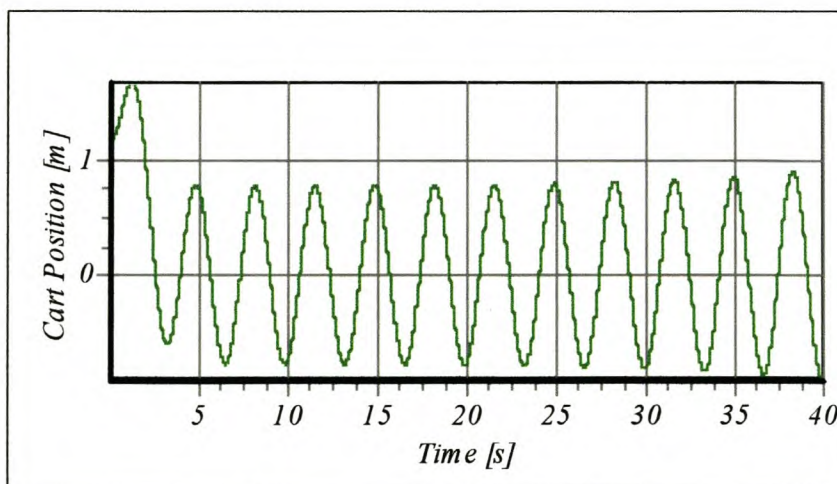


Figure 9-12 - Linear controller transient response for the cart position with a pole length of 0.59m in the nonlinear closed loop.

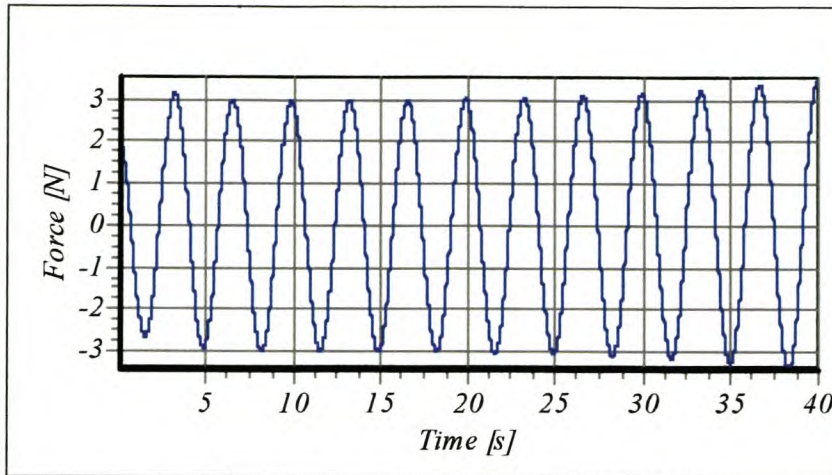


Figure 9-13 - Linear controller manipulated variable control action for a pole length of 0.59m.

The *neurocontroller* maintains an effective *ITAE* response with a pole length increase from 0.5m to 1.13m (226% increase in pole length), as illustrated by figures 9-14 to 9-18. A pole length greater than 1.13m causes the *neurocontroller* to exceed the allowed 20° pole angle deviation in the 40s simulation. The *neurocontroller* thus displays a high level of robustness to parameter uncertainty in the model. The *neurocontroller* has not caused control response to become oscillatory as in the case of the linear controller. The settling time and decay ratio has been maintained as for the nominal pole length. The *neurocontroller* is thus able to maintain stability in the presence of model parameter uncertainty. This is attributed to the ability of neural *networks* to effectively generalise to unseen states in the *environment*.

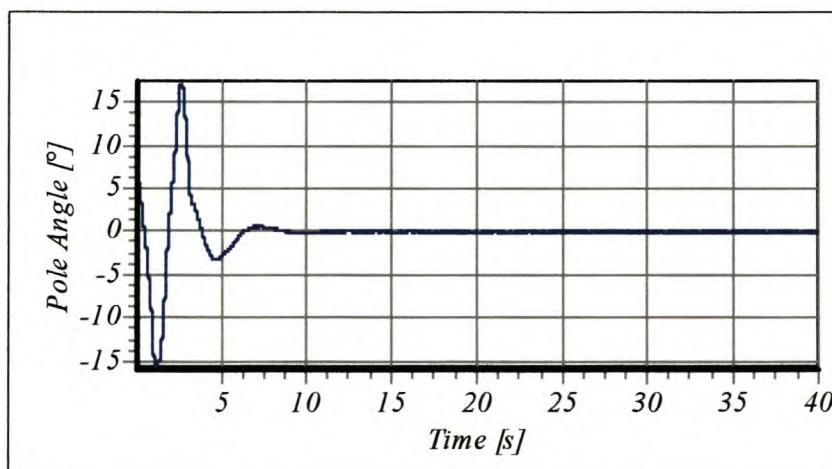


Figure 9-14 - Transient response for the pole angle with a pole length 1.13m.

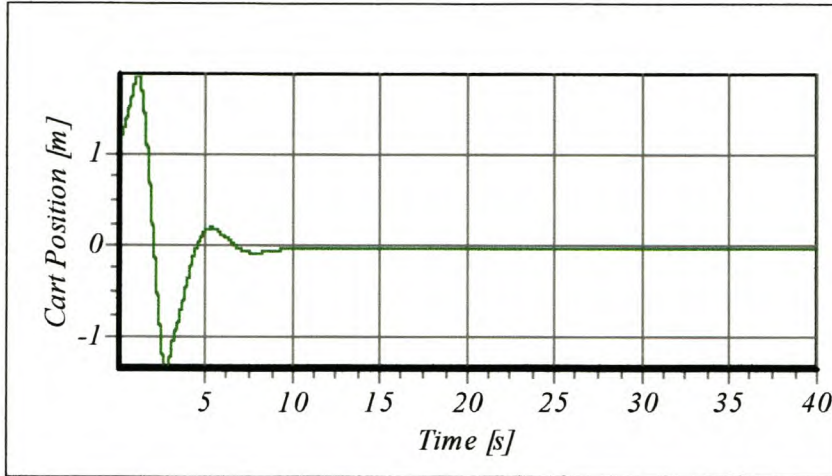


Figure 9-15 - Transient response for cart position with a pole length of 1.13 m.

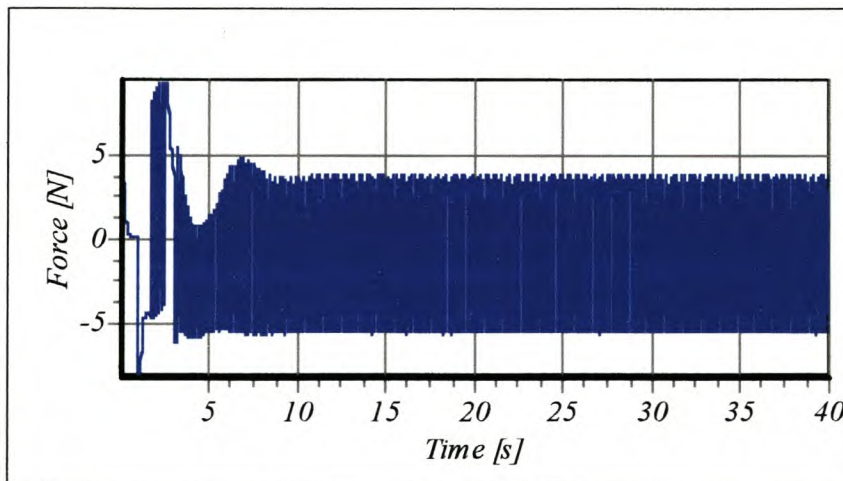


Figure 9-16 - Manipulated variable action with a pole length of 1.13m.

A further benefit of the neurocontrol approach is that anticipated *process/model mismatches* may be effectively incorporated into the controller design procedure, by introducing variable model parameters during learning. Even greater robustness to model parameter uncertainty may thus be afforded. In the *SANE* design approach a model parameter may be considered variable within a specified range during the *evaluation* phase. Linear design techniques do not allow for variable model parameter in the design procedure. Neural *networks* that learn to effectively control over the whole range of the variable model parameter, will thus garner greater *fitness* than neural *networks* that are only effective (specialised) for a narrow variability range. The *SANE* approach thus allows for the development of generalists that are

able to cope with a wide range of model uncertainty. Greater *generalisation* ability may be incorporated into the controller in this manner.

9.4.3 Comparative Control Performance in the presence of disturbances

To simulate a disturbance an external impulse force of +30N is applied to the cart. The linear controller's regulatory performance is illustrated in figure 9-17 and figure 9-18. The responses are extremely oscillatory with a 20s settling time. Increasing the magnitude of the impulse force to +32N causes the linear controller to exceed the 20° allowable operation bandwidth. In contrast, the *neurocontroller's* responses (figure 9-19 & figure 9-20) maintains an *ITAE* characteristic response with a settling time of approximately 5s. High regulatory performance is thus observed for the developed *neurocontroller* in the presence of large disturbances. The magnitude of the impulse force needs to be increased to 67N before the 20° allowable operation bandwidth is exceeded. Effective regulation for external disturbance is thus introduced by the ability of the neural *network* to learn behaviour, that allows it to generalise in the presence of unexpected occurrences in the *environment*. More autonomous control is thus possible as a result of greater robustness to unmeasured disturbances.

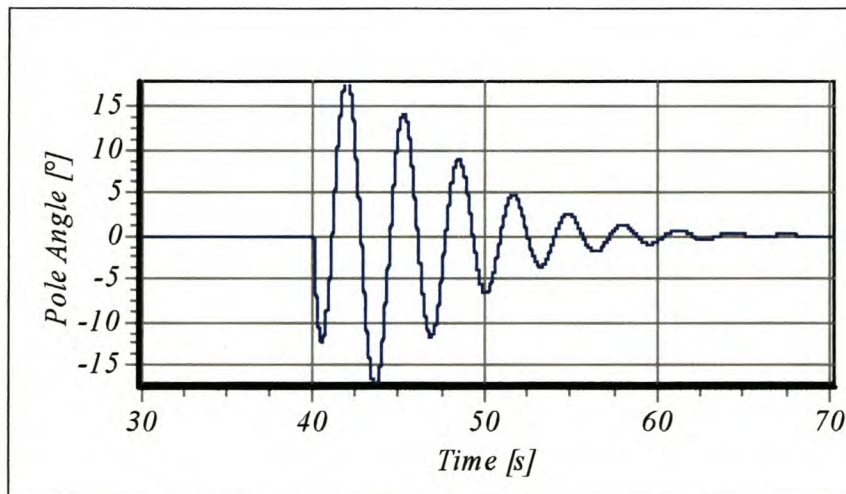


Figure 9-17 - Linear controller pole angle response to an applied force disturbance of +30N after steady state has been reached

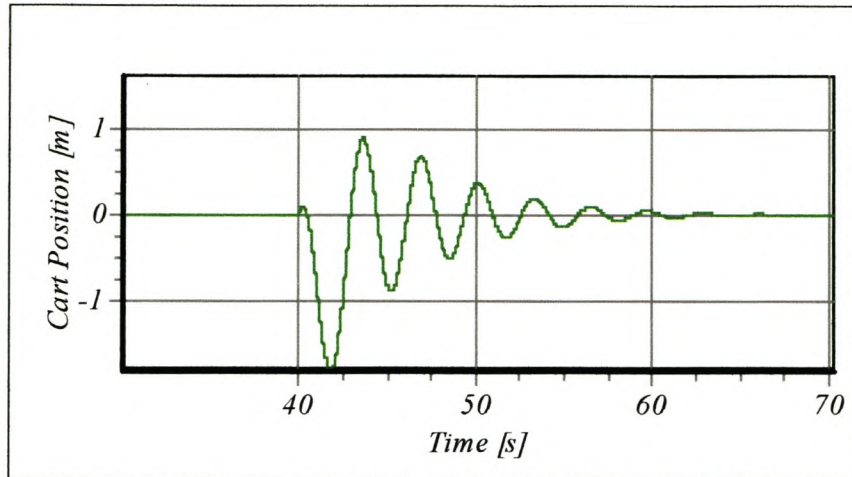


Figure 9-18 - Linear controller cart position response to an applied force disturbance of +30N after steady state has been reached.

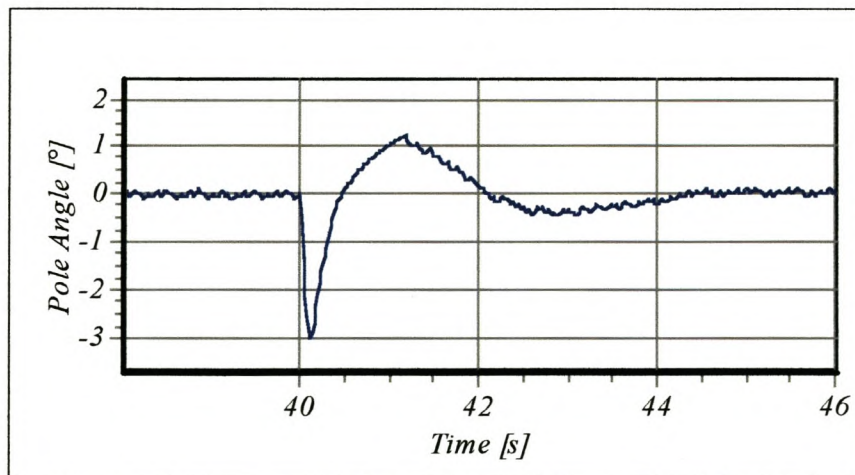


Figure 9-19 - Neurocontroller pole angle response to an applied force disturbance of +30N after steady state has been reached

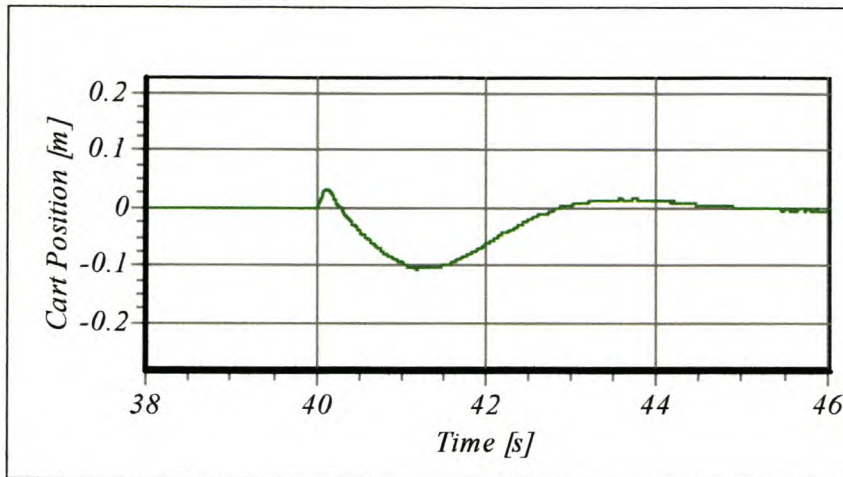


Figure 9-20 - Neurocontroller cart position response to an applied force disturbance of +30N after steady state has been reached.

9.5 SANE LEARNING ROBUSTNESS

9.5.1 Non-Markov Robustness Analysis

As discussed in section 2.6 *non-Markov environments* represent the norm. The learning robustness of the *SANE* algorithm in *non-Markov environments* is thus a paramount consideration to its utilisation in real world *environments*. Such uncertain *environments* may preclude knowledge of the state variables of the system. To learn effective decision policies in *non-Markov environments*, the unmeasured states must be given an internal representation in the developed neural *network* controller. One of the limitations of *SANE* lies in that only feedforward neural *networks* may be developed. No internal representation of temporal state variables (i.e., rate of change state variables) is thus possible, as this would require the development of a recurrent neural *network*. This fact is evident in figure 9-21 where *SANE* is unable to learn a control policy given only the pole angle and the cart position as inputs to the *network*. In 25 generations no policy could be found that could stabilise the unstable system at any *evaluation*. The provided input state representation to the *neurocontroller* is decidedly *non-Markov* as the angular velocity and the cart velocity is not provided as inputs to the *neurocontroller*. This limited availability of state variables may, however, be a realistic design requirement as additional costly sensors, required to

directly provide temporal state variable information, may not form part of a cost effective design.

A solution to the lack of temporal state information, is to provide state estimation through the use of numerical differentiation of past state values. A three point numerical differentiation scheme, utilising the current and past two historical data points, provides sufficient information to develop an effective control *policy*. The effect on the learning rate is, however, marked as is illustrated by a comparison of the full state feedback scenario (figure 9-24) and the temporal state estimation scenario (figure 9-25). For the state estimation scenario a control *policy* that stabilises the pole for the full duration of the trial is only obtained after generation 10, whilst for the full state feedback case a stabilising controller is already obtained from the third generation (same initialisation utilised). This learning delay is attributed to inaccurate state estimation resulting from the numerical differentiation. Although this inaccurate estimation results in a pronounced initial delay in obtaining a stabilising control *policy*, the algorithm learns to deal with discrepancies resulting from state estimation errors and by generation 25, the fittest *individuals* for the full state and estimated state case exhibit similar controller performance. The state estimation case's population average *fitness* is, however, approximately half as fit at generation 25, as the average population for the full state feedback case. This reflects the added difficulty in learning the task with state estimation that may prove inaccurate. The average *population fitness* proves an indication as to what extent stable niches (*specialisations*) have been established in the *population*. Stable equilibrium niche formation is thus retarded for the state estimation scenario. The algorithm, however, remains robust and obtains effective decision policies.

Despite *SANE*'s limitation in not being able to form internal representations of temporal states; it is, however, capable of learning an effect control *policy* in a less non-*Markov environment*. Omitting the cart velocity from the inputs to the *neurocontroller* allows for the development of an effective control strategy, even though that state variable may not be estimated. This results as the dynamics of the pole angle transient is far faster than for the transients of the cart. Being able to effectively learn a *policy* for controlling the pole angle, allows the algorithm to effectively manage without the cart velocity as state input for most initial states.

Should the cart's initial position be close to the boundaries of the track, a loss in robust performance may be inherent, due to the lack of full state information. The pole angular velocity remains a critical temporal state variable, as the fast dynamics for the pole angle transient requires this information for effective control.

As discussed in section 2.6.3 *evolutionary* approaches to *reinforcement learning* posses greater robustness when faced by *non-Markov environments* than *temporal difference* methods (such as *Q-learning*). This may be ascribed to the fact that *policy* changes for *evolutionary* approaches are based on more global information of controller performance, than *temporal difference* methods that update the *policy* based on local information (section 2.9). Global information *evaluation* may allow for the development of policies in which absent state information may be effectively eliminated from consideration.

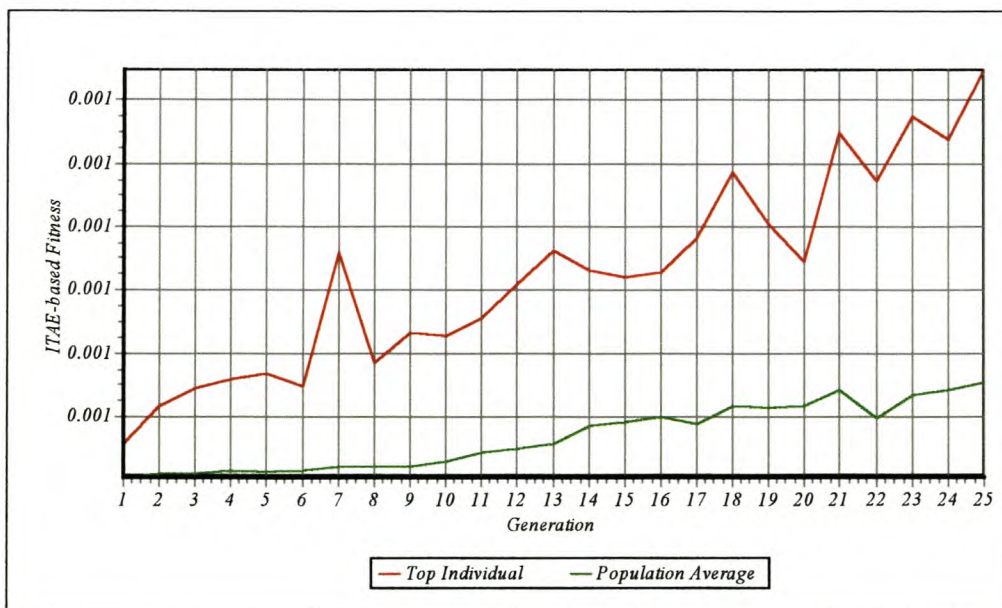


Figure 9-21 - Only Angle and Position data afforded to the neurocontroller as state inputs, resulting in a highly non-Markov state representation to the neurocontroller.

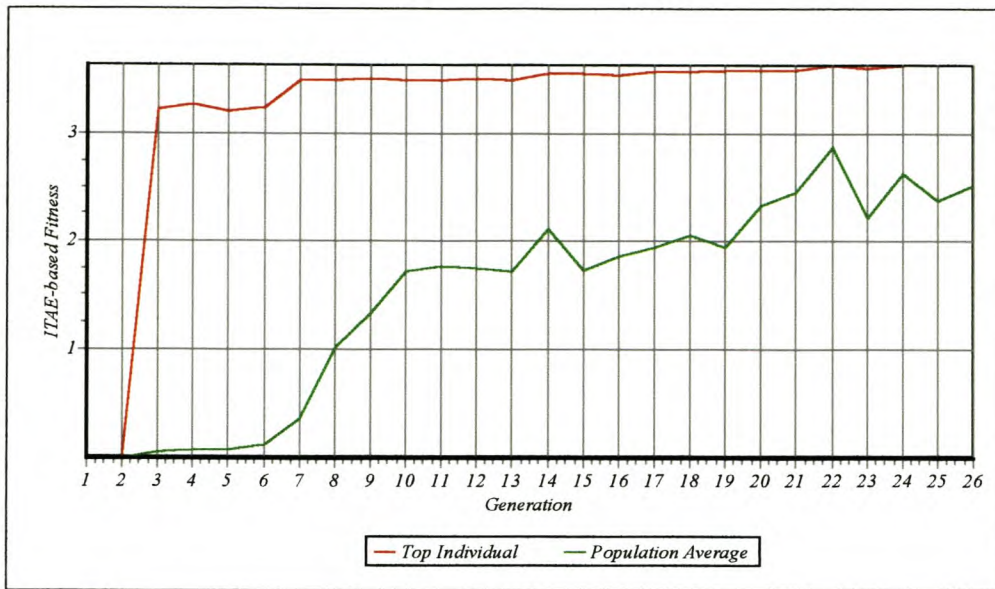


Figure 9-22 - Nominal full state variable feedback scenario.

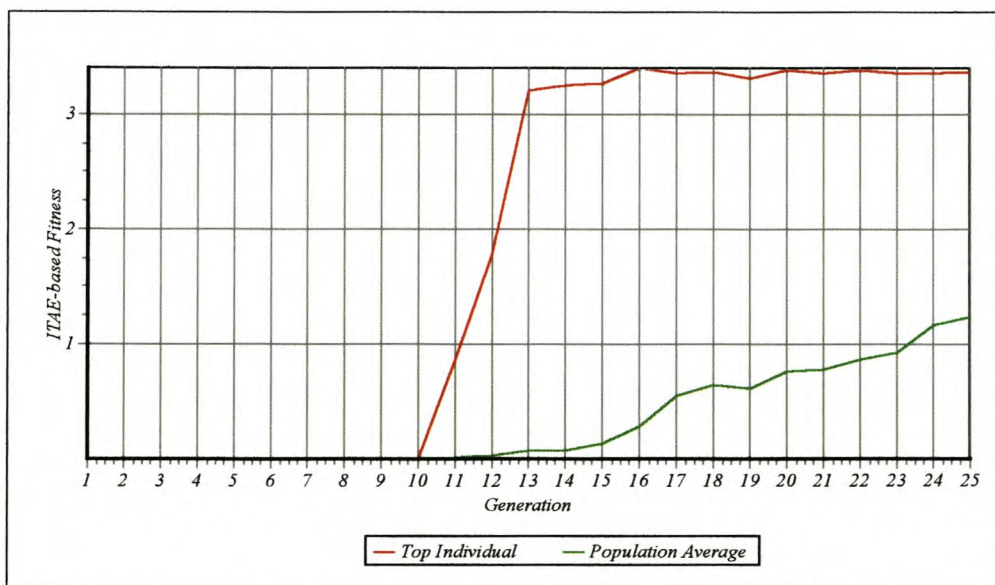


Figure 9-23 - Temporal data obtained by means of state estimation utilising numerical differentiation.

9.5.2 Learning Ability in a Noisy Environment

As real world applications are characterised by sensor noise and inaccurate state *evaluation*, *SANE's* learning rate was evaluated in the presence of *gaussian* noise on all state variables (at each time step), with a standard deviation of 2.5% for the full scale range of each sensor input. The learning rate is illustrated in figure 9-24. This

degree of sensor noise has minimal effect on the learning algorithm. The *ITAE fitness* is not as high as in the full state feedback case (figure 9-22), as the noise on the input data at each time interval results in a continued disturbance to the process. The number of generations to establish stable control of the system is also slightly increased due to the added difficulty of searching for useful behavioural information in the noise filled state space. The *SANE* algorithm is thus able to extract useful information in the presence of misinformation, from its *environment* to form effective control policies.

The utilisation of *evolutionary* strategies for learning in the presence of noise is expected to produce better results than for *temporal difference* methods. As temporal difference methods, such as *Q-learning*, utilise gradient information to update the control *policy*, inaccurate gradient estimation due to noise could produce suboptimal results. The premises of genetic search relies on *schemata* promotion (section 3.1) which does not rely on the determination of gradient information for the state information.

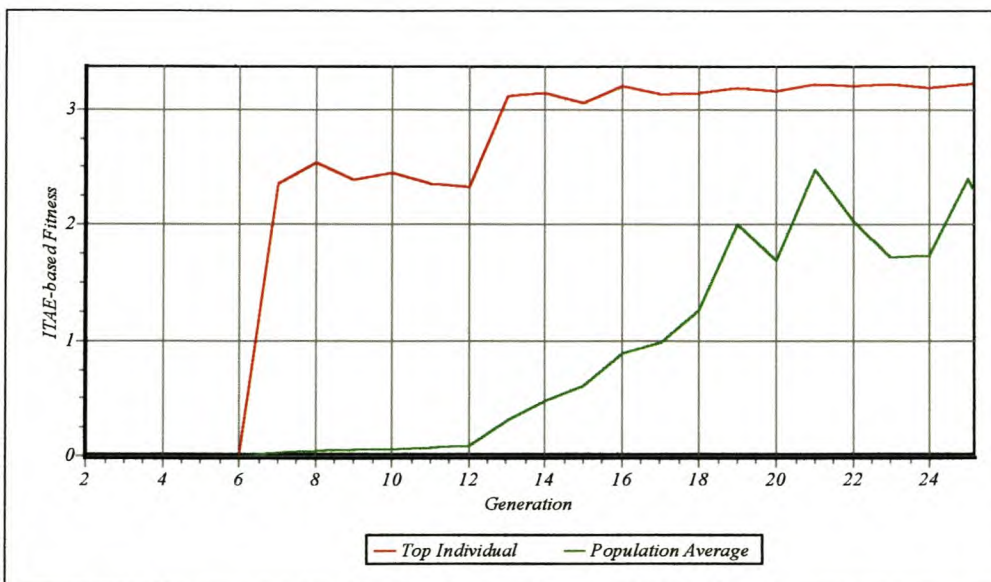


Figure 9-24 - Learning in an environment with sensor noise. Sensor noise is added to the actual state value with a gaussian standard deviation. Angle with noise STD 0.6°. Angular Velocity STD 3°/s. Position STD 0.12. Cart Velocity STD 0.075.

9.6 CONCLUDING REMARKS

This chapter allowed for the development of the methodology used for developing all the *neurocontrollers* in this study; within the framework of an historical *reinforcement learning* benchmark, the inverted pendulum problem. This frequently addressed control problem formed the basis for incorporating the *ITAE* response criteria as *fitness* function into *SANE*. Also, the inverted pendulum problem presented a suitable *environment* in which to test the applicability of *SANE* to process control problems. This applicability analysis included an investigation into the *generalisation* ability of the *neurocontroller* with model uncertainty. The learning rate could also be assessed in a non-*Markov* state representation and in the presence of sensor noise, as may be encountered in real world applications. The comparative analysis between the developed linear feedback controller and the *neurocontroller*, highlighted the fact that nonlinear controller structures will generally outperform linear controllers in nonlinear *environments*.

9.7 SYMBOLS FOR APPENDIX A

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
e	Error from set point	-
F	Applied force	[N]
g	Gravitational acceleration	$[\frac{m}{s^2}]$
J	ITAE minimization criteria	-
l	Pole length	[m]
m	Mass of pole	[kg]
M	Cumulative mass of pole and cart	[kg]
t	time	[s]
u	Control action	[N]
x	Cart position from centre of track	[m]
\dot{x}	Cart velocity	$[\frac{m}{s}]$
\ddot{x}	Cart acceleration	$[\frac{m}{s^2}]$
y	Output state variables	-

Greek symbols

<i>Symbol</i>	<i>Description</i>	<i>Unit</i>
θ	Pole angle	[°]
$\dot{\theta}$	Pole angular velocity	$[\frac{^\circ}{s}]$
$\ddot{\theta}$	Pole acceleration	$[\frac{^\circ}{s^2}]$

10 APPENDIX B

FUNCTIONAL DESCRIPTION OF UTILISED DELPHI UNITS AS INCLUDED ON COMPACT DISC

10.1 SOURCE CODE COMMON TO SIMULATION STUDIES AND REAL WORLD APPLICATION

10.1.1 Best_Nets.pdf

- Structure for a *network individual* in the *blue print population* comprising address pointers to the *neuron population*.

10.1.2 PointNeuronPop.pdf

- Data structure for a *population* of pointers to *neurons* in the *neuron population*.

10.1.3 Sane_Config.pdf

- Configuration data for the *evolutionary* process such as the number of hidden nodes, number of input nodes and number of output nodes.

10.1.4 Sane_Domain.pdf

- Provides for the *evaluation* of a single *neurocontroller* from the *network population* and returns the *fitness* value of the *evaluation*.

10.1.5 Sane_EA.pdf

- Quicksort to rank both the *neuron* and *network populations* after *evaluation*.

- Facilitates the application of genetic *operators*, one point *crossover* and *mutation*, to the *neuron* and *network populations*.
- Logs the genetic information for both the *neuron* and the *network population* to a Paradox 7.0 database.

10.1.6 Sane_Network.pdf

- Data structure for a neural *network (neurocontroller)*.

10.1.7 Sane_Networkpop.pdf

- Data structure for a *population* of address pointers to neural networks (*neurocontrollers*).

10.1.8 Sane_Neuron.pdf

- Data structure for a *neuron* in the *neuron population*.

10.1.9 Sane_Neuronpop.pdf

- Data structure for a *population* of *neurons*.

10.1.10 Sane_NN.pdf

- Provides for the decoding and *encoding* of *neurocontrollers*.
- Calculates the neural *network* output for a given neural *network* input.

10.1.11 Sane_Util.pdf

- Allows, together with BestNetThread.pas, for testing the best *neurocontroller* in the closed loop.

- Evaluates the entire *network population*.
- Allows for the *neuron* and *network populations* stored in database tables to be loaded into memory.

10.1.12 SaneAlgThread.pdf

Provides a secondary thread (multi-threading) of the *evolutionary* process.

10.2 SOURCE CODE UNIQUE TO MEBAD MAN_MACHINE INTERFACE AND SANE ON-LINE LEARNING IMPLEMENTATION

10.2.1 Access.pdf

- Manages the input of *process variable* data through analogue-to-digital conversion from the Advantec 812PG card.
- Logs all process data and observations (in form of notes) to a Paradox 7.0 database.
- Provides real time value of any *process variable* and trends historical data for a predefined number of data points for up to two *process variables* simultaneously. This is accomplished through a user-friendly Windows Explorer type interface.
- Utilises nonlinear median filter to remove sensor noise from analogue input data.
- Manages digital output to solid state relay for heating mantle control.
- Manages the input of *process variables* to the PI control routine and the analogue output to the variable speed drives (reflux pumps).
- Determines whether PI control needs to be activated during an *evaluation* of a *neurocontroller*.
- Plots a temperature profile across the entire column.

10.2.2 AdvanAO.pdf

- Used in access.pas to facilitate analogue output (0-10V) from the Advantec 812PG to the variable speed drives.

10.2.3 AIFilter.pdf

- Used in access.pas to convert multiple channels of analogue input data to digital data.
- Used to set the sample rate for analogue-to-digital conversion.
- Filters the input data using a 5 sample point nonlinear median filter.

10.2.4 BestNetThread.pdf

- Facilitates the testing of the best *neurocontroller* for the open *population* for an indefinite amount of time via a secondary thread.

10.2.5 Driver.pdf

- Functions provided by Advantec Inc. to communicate with the 812PG card from a Win32 application.

10.2.6 Global.pdf

- Contains globally declared variables.

10.2.7 MantleControl.pdf

- Allows for the heat duty control to the reboiler by providing for a digital output signal to the solid state relay.

10.2.8 PIDControl.pdf

- Allows for PI control calculations.

10.3 SOURCE CODE UNIQUE TO SIMULATION STUDIES

10.3.1 Cstr.pdf / Bioreact.pdf / Pole.pdf

- Calculates the dynamic responses for the simulation studies by Runge-Kutta integration of the equations of state.

11 APPENDIX C

GLOSSARY

<i>Terms</i>	<i>Description</i>
ANN	Artificial Neural Network
Action, control	Control output by the neurocontroller to the manipulated variable, which influences the state of the dynamic environment.
Agent	See reinforcement learning agent
Blue print population	Layer of neural networks (effective neuron collections) that is evolved with the neuron population.
Credit assignment	The challenge of distributing credit for success or failure among many control actions taken by the agent.
Children	Produced after applying the crossover operator to two parents. Two offspring are produced designated as children.
Convergence	Convergence of the genetic population to a single "type" of individual, thereby exhausting the genetic diversity in the population.
Competition (competing species)	Competition between individual in the population to perform a similar function (specialisation) in the environment (weak cooperation)
Cooperation (cooperating species)	The division of the various subtasks between various species in the population that represent the complete solution.
Crossover	Genetic operator whereby two parents exchange genetic material at a break point to form two children.
Diversity	A measure of the amount of unique genetic material available in the population to drive the genetic search.
Domain	See environment
Elite	The top individuals in the neuron and network that are subject to crossover.
Encoding	Genetic string encoding (real value) that represents the necessary weight and neuron connections.
Environment	Dynamic system in which the agent is able to perceive the system state and change the system's state via a control action.

Evaluation stage	Allows for the determination of both network and individual neuron fitness values, by evaluating each agent's performance in the environment.
Evolutionary methods (approaches)	Probabilistic optimisation methods based on assigning more evaluations to genes that raise the average fitness of the population.
Exploitation	The agent selects an action(s), that from past learned experience, has resulted in increased reward.
Exploration	The agent selects an action(s) that it has not attempted in the past, in order to discover new useful actions to obtaining the maximum reward.
Fitness	The criteria that drives the genetic search to a particular solution. A fitness value (reward) is assignment to each agent (individual) during the evaluation stage.
Gaussian disturbance (noise)	Process variable or model paramater is varied around a mean with a gaussian distribution.
Gene	Section of the individual (agents) encoding that represents either a network weight or a connectin assignment.
Genetic algorithm	Genetic search (optimisation) method that promotes the propagation of genes in the population that allow for above average performance of the individuals in the population.
Generalisation tools (methods)	Allows for the sharing of knowledge between similar states (conditions) in the environment, thereby allowing for a compact representation such as in a neural network's parallel structure.
Genotype	The encoded representation of an individual, either a neuron or a network.
Hyperplanes	Sections in the hyperspace that share similar genetic information across a plane in the multi-dimensional solution space.
Hyperspace	Multi-dimensional representation of all possible solutions in the solution space.
IMC, Internal Model Control	The forward and inverse models of the plant (process) are used directly in the control structure to determine an appropriate control action.
Implicit fitness sharing	Pertains to the search for cooperative and competing individual in the population that perform the necessary number of subtasks, which make up the complete solution.
Individual	An encoded neuron or network in the genetic population that is subjected to genetic operators by the genetic algorithm.

ITAE criteria (response)	Integral Time Absolute Error. Integral of absolute error multiplied by the time. Response characterised by some overshoot with insignificant oscillation.
Manipulated variable	The final control element, i.e. control valve, variable speed drive
Markov property	The assumption that the agent, at each time instant, has access to all state information
MEBAD	Multi-Effect Batch Distillation Configuration
MIMO (Multi-Input-Multi-Output)	Single control system containing multiple process variables (input) and multiple manipulated variables (output)
Model Mismatch Compensation	Technique used in MPC to adjust inaccurate model parameters to reflect the actual process parameters, thereby reducing steady state offset.
MPC, Model Predictive Control	The control structure incorporates a model of the process, which an optimisation routine (controller) utilises to determine appropriate control actions for a number of time steps into the future.
Mutation	Genetic operator by which a gene (weight or connection) is reassigned asexually with a given (small) probability.
Network, neural feedforward	Neural network in which information is propagated in one direction - from the input layer, to the hidden layer, to the output layer.
Neuron	Basis building unit of a neural network, which represents connections and their associated weights from the hidden to the input and output layers.
Neurocontroller	Neural network that performs the control function in the closed loop.
Niching	An emergent phenomena in which an evolving population of neurons perform distinct subtasks (specialisations) that form a complete solution.
Nonminimum phase behaviour	Characterised by a process variable initially having an inverse response to an appropriate manipulated variable control action.
Offspring	See children.
Operators, genetic	Crossover and mutation.
Optimal behaviour, models of	Determines how the agent should take the future into account with the actions it is taking in the present.
Parents	The two elite individuals selected from the population to participate in crossover.

Process / model mismatch	Discrepancies between the developed dynamic model of a process utilised in controller development and the actual process.
Process variable	Process state utilised in determining a control action.
Policy	The behaviour that an agent has developed through learning, with which it executes the desired task.
Population	The entire collection of individuals (neurons or networks) that are used by the genetic algorithm to search for the optimal solution.
Q-learning	Reinforcement learning technique that progresses toward an optimal solution via a temporal difference method.
Recombination stage	Involves the application of genetic operators to the neuron and network populations. Elite individuals undergo crossover, while the remainder of each population is mutated.
Reinforcement learning	A computational approach in which an agent learns through trial-and-error interaction with a dynamic environment to perform a specific task, without being presented with how the task should be performed.
Reinforcement learning agent	A neural network (individual in the genetic population) which learns through interaction with its environment to perform a given task in its environment.
Reward	A scalar value assignment to each agent based on its ability to perform the desired function (fitness value).
SANE	Symbiotic, Adaptive Neuro Evolution. A genetic algorithm which utilised implicit fitness sharing to evolve neural networks that perform a specified task in a dynamic environment.
Schemata promotion	The propagation (increase) of those genes that contribute to an individual's success in the dynamic environment throughout the population via crossover.
SISO (Single-Input-Single-Output)	Typical PID control configuration, in which a single process variable is used to calculate a single control action.
Specialisation (Speciation)	See niching.
State representation	The collection of state variables that describes the dynamic condition of the process at any given moment in time.
State variables	Those process variables, that once the dynamic equations describing them are integrated, give a representation of the dynamic state of the process.

Steady State Offset	The error between the final process steady state and the desired set point.
Temporal difference methods	Gradient based search technique used to minimise the error from the desired state in reinforcement learning environments.
Transformation (transition)	The function performed by a neural network which calculates the control action based on the current plant state.
Trial	See evaluation
Value	The true value of a control action in a particular environmental state taking the future usefulness of the action into account; not only the immediate reward
Weight	Determines the value of its associated connection (preceding neuron) in determining the neuron's output (activation)